Digital Electronics - Théorie

Pol Nicolaï - Guilian Deflandre - François Custinne
mis à jour par Louis Groulard

15 mai 2019

1 Chapitres 1,2 et 3

1.1 Question 1

Quelle est la différence entre un système analogique et un système numérique ?

Un signal analogique se base sur un signal continu $\left(\frac{amplitude}{temps}\right)$. Un système numérique se base lui sur un signal discrétisé. La différence est donc que les systèmes numériques manipulent des valeurs discrètes alors que les systèmes analogiques manipulent des valeurs continues.

1.2 Question 2

La fonction NAND/NOR/XOR/NXOR est-elle associative ?

Les fonctions XOR et NXOR sont associatives.

1.3 Question 3

Qu'est-ce que le minterme d'une fonction booléenne ? Donnez un exemple.

On appelle minterme le produit de degré n dans lequel toutes les variables de F ou leurs formes complémentées sont présentes.

❑ *Example*: n = 3.

X	Y	Z	Product Term	Symbol	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	0	0	$\overline{X}\,\overline{Y}\,\overline{Z}$	m_0	1	0	0	0	0	0	0	0
0	0	1	$\overline{X}\,\overline{Y}Z$	m_1	0	1	0	0	0	0	0	0
0	1	0	$\overline{X}Y\overline{Z}$	m_2	0	0	1	0	0	0	0	0
0	1	1	$\overline{X}YZ$	m_3	0	0	0	1	0	0	0	0
1	0	0	$X\overline{Y}\,\overline{Z}$	m_4	0	0	0	0	1	0	0	0
1	0	1	$X\overline{Y}Z$	m_5	0	0	0	0	0	1	0	0
1	1	0	$XY\overline{Z}$	m_6	0	0	0	0	0	0	1	0
1	1	1	XYZ	m_7	0	0	0	0	0	0	0	1

1.4 Question 4

Qu'est le maxterme d'une fonction booléenne ? Donnez un exemple.

On appelle maxterme une somme de degré n dans laquelle toutes les variables de F ou leurs formes complémentées sont présentes.

❑ Example: n = 3.

X	Y	Z	Sum Term	Symbol	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	$X + Y + Z$	M_0	0	1	1	1	1	1	1	1
0	0	1	$X + Y + \overline{Z}$	M_1	1	0	1	1	1	1	1	1
0	1	0	$X + \overline{Y} + Z$	M_2	1	1	0	1	1	1	1	1
0	1	1	$X + \overline{Y} + \overline{Z}$	M_3	1	1	1	0	1	1	1	1
1	0	0	$\overline{X} + Y + Z$	M_4	1	1	1	1	0	1	1	1
1	0	1	$\overline{X} + Y + \overline{Z}$	M_5	1	1	1	1	1	0	1	1
1	1	0	$\overline{X} + \overline{Y} + Z$	M_6	1	1	1	1	1	1	0	1
1	1	1	$\overline{X} + \overline{Y} + \overline{Z}$	M_7	1	1	1	1	1	1	1	0

1.5 Question 5

Quelle est la relation entre un minterme m_i et le maxterme correspondant M_i ? Donner un exemple pour une expression à 3 variables binaires.

Le maxterme est le complément du minterme. C'est-à-dire que

$$M_i = \overline{m_i} \quad \text{et} \quad m_i = \overline{M_i}$$

En effet, si on prend le maxterme $M_0 = X + Y + Z$, le minterme correspondant sera $m_0 = \overline{X}\,\overline{Y}\,\overline{Z}$. Ceci peut être facilement démontré avec les lois de DeMorgan.

$$\overline{M_0} = \overline{X + Y + Z} = \overline{X}\,\overline{Y}\,\overline{Z} = m_0$$

1.6 Question 6

Qu'appelle-t-on un implicant d'une fonction booléenne ?

Un implicant est un produit d'une fonction Booléenne si il a la valeur "1" pour tous les minterms du produit.(i.e tous les rectangles sur une K-map fait de carré contenant la valeur 1 correspondent à des implicants).

1.7 Question 7

Qu'appelle-t-on l'impliquant premier d'une fonction booléenne ?

Un impliquant premier est un produit obtenu en combinant le plus grand nombre "2^k" possible de mintermes adjacents de sorte que ce produit ne peut plus être simplifié.

1.8 Question 8

Qu'appelle-t-on l'impliquant premier essentiel d'une fonction booléenne ?

Un impliquant premier est essentiel si il contient un minterme qui n'est inclus dans aucun autre impliquant premier. La suppression de cet impliquant ne permet donc plus de couvrir tous les mintermes de la fonction.

1.9 Question 9

Qu'est ce qu'une fonction booléenne paire/impaire ? Donnez son expression sous forme d'une somme de produits et la table de vérité correspondant dans le cas particulier de la fonction paire/impaire à 4 variables

---

La fonction impaire est égale à 1 si elle possède un nombre impair d'entrées égales à 1 = XOR.

- L'exemple à 4 variables binaires est donné par

$$A \oplus B \oplus C \oplus D = A\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,C\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,D$$

- La table de vérité est la table de vérité d'un XOR classique à 4 variables

La fonction paire est égale à 1 si elle possède un nombre pair d'entrées égales à 1 = XNOR.

- L'exemple à 4 variables binaires est donné par

$$\overline{A \oplus B \oplus C \oplus D} = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\,C\,D + A\,\overline{B}\,C\,D + A\,B\,\overline{C}\,D + A\,B\,C\,\overline{D}$$

- La table de vérité est la table de vérité d'un NXOR classique à 4 variables.

1.10 Question 10

Citez une application importante de fonction impaire ?

- Additionneurs, soustracteurs
- Compteurs
- Test de parité et générateur des bit de parité

1.11 Question 11

Qu'est ce qu'un bit de parité ?

Un bit de parité est le bit rajouté qui va déterminer si le nombre total de bits de la variable est pair ou impair. le 0 qui mit le nombre de pair ou impair

1.12 Question 12

Qu'est ce qu'un circuit intégré ?

On implémente des circuits en utilisant des transistors et des interconnexions au sein de semi-conducteurs. Ces semi-conducteurs sont appelés circuits intégrés.

1.13 Question 13

Qu'est ce que la marge de bruit (noise margin) d'une porte logique ?

C'est la tension de bruit externe maximale superposée à la valeur d'entrée normale qui ne provoquera pas de changement indésirable dans la sortie du circuit.

1.14 Question 14

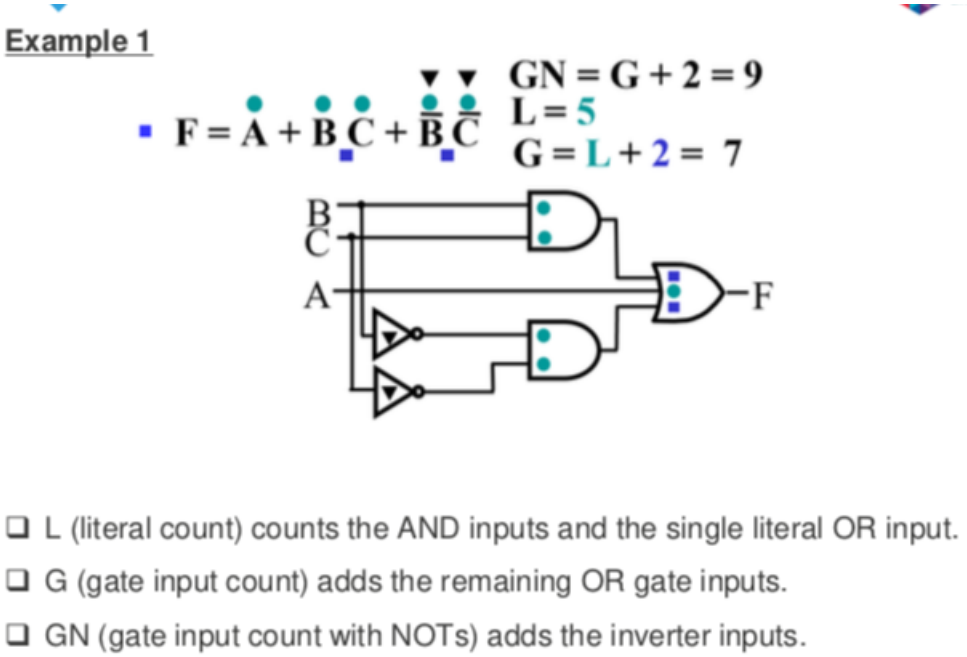
Qu'est ce qu'un code de Grey ?

Le code de Grey est un type de codage binaire permettant de ne modifier qu'un seul bit à la fois quand un nombre est augmenté d'une unité. Cette propriété est importante pour plusieurs applications.

1.15 Question 15

Qu'appelle-t-on le *literal cost* ? Donnez un exemple.

Le *literal cost* est le nombre d'apparition littérales dans une expression booléenne correspondant au schéma de circuit logique.

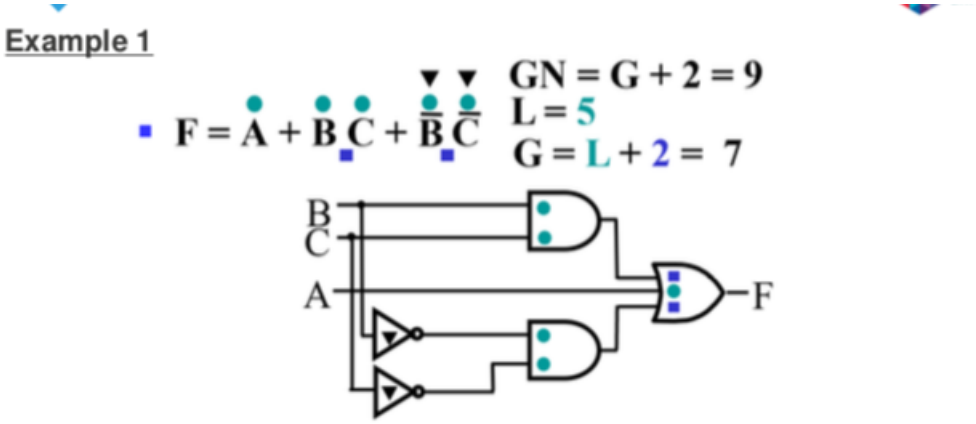


- L (literal count) counts the AND inputs and the single literal OR input.
- G (gate input count) adds the remaining OR gate inputs.
- GN (gate input count with NOTs) adds the inverter inputs.

1.16 Question 16

Qu'appelle-t-on le *gate input cost* ? Donnez un exemple.

Le *gate input cost* est le nombre d'entrées aux portes logiques dans l'implémentation correspondant exactement à l'équation ou aux équations données



- ❑ L (literal count) counts the AND inputs and the single literal OR input.
- ❑ G (gate input count) adds the remaining OR gate inputs.
- ❑ GN (gate input count with NOTs) adds the inverter inputs.

2 Chapitre 4 et 7

2.1 Question 17

Qu'est-ce qu'un circuit combinatoire ?

Un circuit combinatoire :

- est activé par n signaux d'entrée numérique = n variables booléennes $e_1, ..., e_n$
- fournit n signaux de sortie numérique = m variables booléennes $s_1, ..., s_n$
- est constitué de n circuits logiques, chacun réalisant une fonction logique $s_j = F(e_1, ..., e_n)$, $j = 1, ..., n$

2.2 Question 18

Qu'est-ce qu'un décodeur (decoder) en logique combinatoire ? Expliquer brièvement son fonctionnement.

Un décodeur permet la conversion d'une entrée à n bits en une sortie à m bits avec $n \leq m \leq 2^n$ et telle que chaque code d'entrée produit une sortie unique. Ainsi, un décodeur n-to-m va générer m mintermes pour les n variables d'entrées. Pour une entrée à 2 variables, le décodeur aura donc la table de vérité suivante.

A_0	A_1	D_4	D_3	D_2	D_1
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

2.3 Question 19

Qu'est-ce qu'un encodeur (encoder) en logique combinatoire ? Expliquer brièvement son fonctionnement.

Un encodeur fonctionne de la manière inverse d'un décodeur. Il s'agit donc de la conversion d'une entrée à m bits en une sortie de n bits avec $n \leq m \leq 2^n$ et telle que chaque code ~~d'entrée~~ produit une ~~sortie~~ unique. Pour un encodeur à 8 entrées, l'encodeur aura la table de vérité suivante. Le reste de la table de vérité fournit une sortie où chaque bit vaut 0.

Décimal	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0	0	1
2	0	0	0	0	0	1	0	0	0	1	0
3	0	0	0	0	1	0	0	0	0	1	1
4	0	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	0	0	0	0	1	0	1
6	0	1	0	0	0	0	0	0	1	1	0
7	1	0	0	0	0	0	0	0	1	1	1

2.4 Question 20

Qu'est-ce qu'un encodeur de priorité (priority encoder) en logique combinatoire ? Expliquer brièvement son fonctionnement.

Le problème de l'encodeur précédent est que si plus d'une des variables d'entrée est égale à 1, il ne fonctionne alors pas. Un encodeur de priorité est donc un encodeur qui peut accepter toutes les combinaisons de valeur d'entrée et produire un résultat logique. Il sélectionnera toujours la valeur avec la position la plus (ou la moins) forte. Il sortira alors le code correspondant à cette position. La table de vérité d'un tel encodeur à 4 entrées est la suivante.

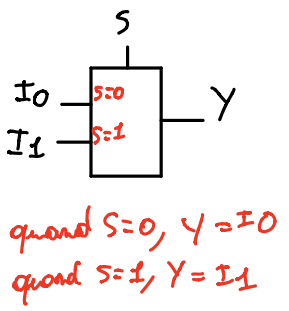
D_3	D_2	D_1	D_0	A_0	A_1	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

2.5 Question 21

Qu'est-ce qu'un multiplexeur (multiplexer) en logique combinatoire ? Expliquer brièvement son fonctionnement.

Un multiplexeur permet de sélectionner de l'information d'une entrée et de contrôler l'information qui sort. Un multiplexeur possède en général n entrées de contrôles, 2^n entrées d'informations et une sortie$y$. Pour un multiplexeur avec 2 entrées d'informations, on a la table de vérité suivante :

S	I_1	I_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



2.6 Question 22

Qu'est-ce qu'un *half-adder* en logique combinatoire ?

Un *half-adder* est un circuit arithmétique à 2 entrées et 2 sorties qui effectue l'addition des 2 bits d'entrées. La somme est exprimée sous la forme d'un bit de somme S, et d'un bit de carry C. On obtient donc la table de vérité reprise à la page suivante.

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Ceci nous permet donc d'obtenir les fonctions :

$$S = X \oplus Y, C = XY$$

2.7 Question 23

Qu'est-ce qu'un *full-adder* en logique combinatoire ?

Le *full-adder* est un bloc fonctionnel comme le *half-adder*, mais possédant 3 entrées et 2 sorties. Le *full-adder* effectue l'addition des 3 bits d'entrée, parmi lesquels se trouve un carry bit retenant l'état précédent. Il possède la table de vérité suivante :

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

2.8 Question 24

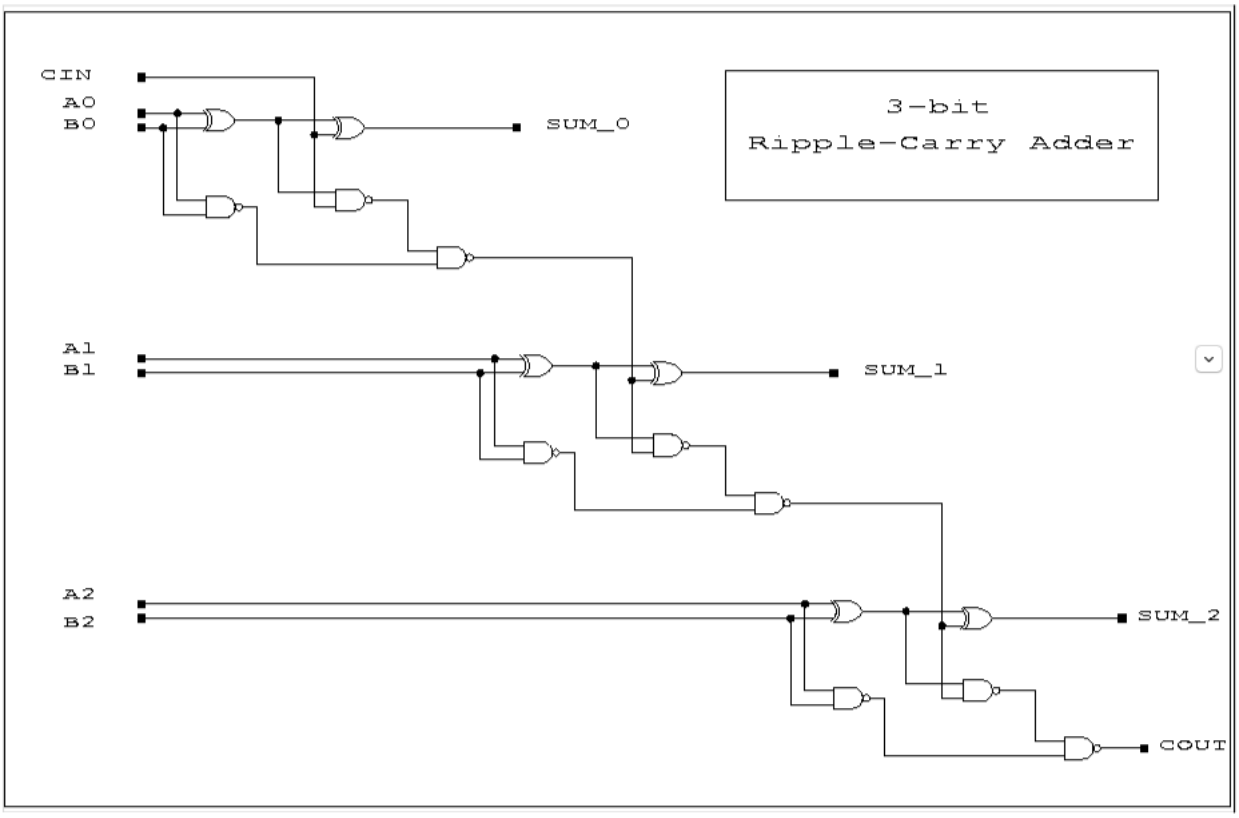
Comment détecte-t-on l'*overflow* d'un additionneur binaire ?

Un *overflow* survient si $n+1$ bits sont nécessaires pour contenir le résultat d'une addition de n bits. Dans le cas d'un additionneur sans signe, un *overflow* sera remarqué lorsque $C = 1$. Dans le cas d'un additionneur signé, un *overflow* ne peut survenir que si les 2 termes de l'addition sont de même signe (et donc leur résultat également). Dans ce cas-là, l'*overflow* surviendra si $V = C_n \oplus C_n - 1$ vaut 1.

2.9 Question 25

Qu'est-ce qu'un *Ripple Carry Adder* ?

Un *Ripple Carry Adder* est un circuit numérique qui produit la somme arithmétique de deux nombres binaires. Il peut être construit avec des *Full-adders* connectés en cascade, en connectant la sortie C de chaque additionneur à l'entrée du prochain additionneur en cascade de la chaîne. En exemple, le circuit d'un 3-bit *Ripple Carry Adder* est donné par le schéma suivant.



2.10 Question 26

Expliquez le fonctionnement d'un soustracteur (substractor).

Un soustracteur est un circuit capable de faire la soustraction de deux nombres binaires d'un bit chacun. Le circuit aura deux entrées X et Y et deux sorties S et B. Un soustracteur fonctionne de la même manière qu'un *adder*, mais en utilisant la technique du complément à 2 pour désigner les chiffres négatifs.

2.11 Question 27

Expliquez comment on utilise le *2s complement* dans un soustracteur (substractor).

La soustraction de deux "n-digit" nombres non-signés, M-N, en binaire peut se faire comme suit :

- ajouter le complément à deux du soustrayant N au termes qu'on soustrait M. Cela veut dire M + (2^n - N) = M - N + 2^n.
- Si M $\geq$ N, la somme produit un repport. Ne pas tenir du repport final, laissant le résultat M - N.
- Si M $<$ N, la somme ne produira pas de repport final, puisqu'elle est égale à 2^n - (N -M), le complément à deux de N - M. On doit donc faire une correction, on prends le complément à deux de la somme et on place un signe moins devant pour obtenir -(N - M).

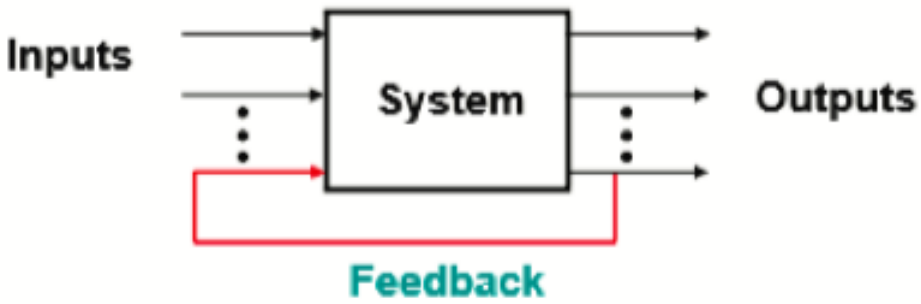
Un exemple est disponible chapitre 7 slide 20.

3 Chapitres 4,6 et 8

3.1 Question 28

Qu'est-ce qu'un circuit séquentiel ?

Les circuits séquentiels sont des circuits qui possèdent de la mémoire. C'est la différence principale avec les circuits combinatoires, qui n'en contiennent pas. Ainsi, la sortie des circuits combinatoires ne dépend donc pas que de la valeur actuelle de ses signaux d'entrée, mais aussi des valeurs de ses signaux précédents. La mémoire est stockée dans ce qu'on appelle un état. On peut représenter ces circuits comme suit :



3.2 Question 29

Qu'est-ce qu'un circuit séquentiel synchrone/asynchrone ?

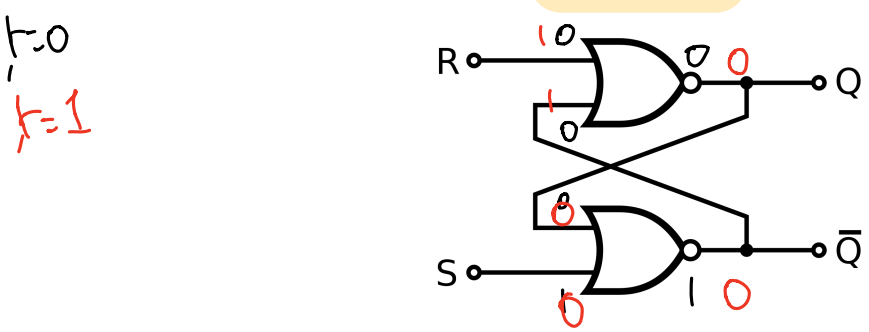
Un circuit séquentiel est synchrone lorsque tout ses éléments ont la même horloge. Ainsi, les changements du système s'opèrent à la même fréquence. Le circuit est à l'inverse asynchrone lorsque l'état du système dépend de plusieurs horloges.

3.3 Question 30

Qu'est-ce qu'un verrou (latch) ? Quelle est la spécificité d'un verrou D (D latch) ?

Un verrou est un circuit logique capable de maintenir une valeur de sortie malgré un changement d'état des valeurs d'entrées. Ce circuit permet de passer d'une logique combinatoire à une logique séquentielle. Il existe 3 type de verrous :

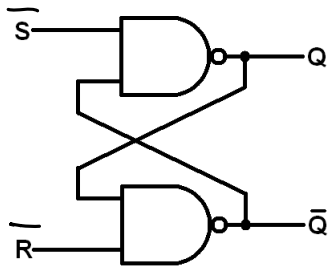
1. Le circuit du verrou SR composé de 2 portes NOR :



On peut décrire son fonctionnement de la manière suivante :

- S (Set) passe à 1 : la sortie Q passe à 1.
- R (Reset) passe à 1 : la sortie Q passe à 0.
- $R = S = 0$ (Memory State) : la sortie Q retient son état précédent.
- $R = S = 1$: l'état est interdit.

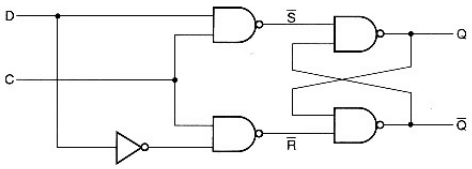
2. Le circuit du verrou $\bar{S}\bar{R}$ composé de 2 portes NAND :



On peut décrire son fonctionnement de la manière suivante :

- S (Set) passe à 0 : la sortie Q passe à 1.
- R (Reset) passe à 0 : la sortie Q passe à 0.
- $R = S = 1$ (Memory State) : la sortie Q retient son état précédent.
- $R = S = 0$: l'état est interdit.

3. Le circuit du verrou D composé de 4 portes NAND et un inverseur :



On peut décrire son fonctionnement de la manière suivante :

- C passe à 0 (Memory State) : la sortie garde son état précèdent.
- C passe à 1 (Update) : la sortie Q prend l'état de D.

La particularité du verrou D est qu'il ne possède d'une entrée de contrôle, D dans le circuit ci-dessus. Notons que le circuit utilise tout de même une entrée de donnée, C sur le schéma.

3.4 Question 31

Qu'est-ce qu'un *flip-flop* (bascule) ?

Un *flip-flop* est un circuit permettant de retenir un état en fonction d'un signal d'horloge. Le *flip-flop* a un comportement synchrone au signal d'horloge, en d'autres termes la valeur de sortie de ce circuit changera uniquement lors d'une variation de ce signal d'horloge.

3.5 Question 32

Quelle est la principale différence entre un verrou et un *flip-flop* ? Quel est l'avantage du *flip-flop* sur le verrou ?

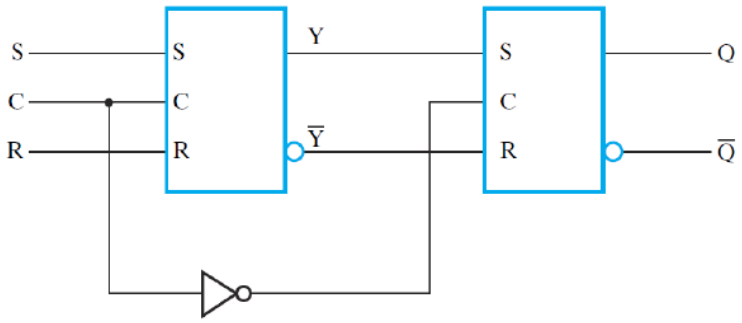
La différence entre le verrou et le *flip-flop* réside dans le fait qu'un *flip-flop* dépend d'un signal d'horloge et possède donc une notion temporelle. En-effet, même si les entrées du *flip-flop* varient, le changement de la sortie ne se fera qu'au flan d'horloge montant ou descendant (suivant le circuit) suivant. Le verrou en revanche dépend uniquement des valeurs prisent par ses entrées. Dés lors, le *flip-flop* à l'avantage de ne pas être sensible à diverses états, dans l'éventualité où ils se produiraient entre 2 flancs d'horloge.

3.6 Question 33

Qu'est-ce qu'un *Master-Slave flip-flop* ?

Le "*Master-Slave flip-flop*", par opposition au "*Edge-Triggered flip-Flop*" [1], est un flip-flop pour lequel le changement d'état du circuit est déclenché par la valeur (haute ou basse) de l'horloge.

Le "*Master-Slave flip-flop*" est composé de deux verrous SR connectés en cascade. De plus, la *clock* est inversée entre le premier et le deuxième verrou. On le représente par :



Son fonctionnement s'organise de la sorte :

- **Quand $C = 1$** :

  51 Le premier verrou est le **maître** et est en mode **transparent**. C'est-à dire qu'il renvoi directement une sortie Y .

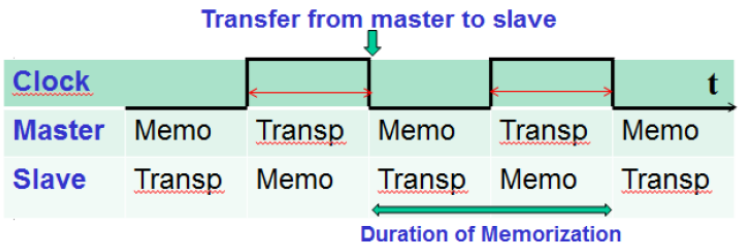
  51 Le deuxième verrou est l'**esclave** et est en mode **mémorisation**. C'est-à-dire que l'état précédent est mémorisé et que l'état actuel correspondant à Y ne pourra pas passer vers la sortie Q .

- **Quand $C = 0$**

  51 Le premier verrou est l'**esclave** et est en mode **mémorisation**, c'est-à-dire qu'aucune valeurs de S ou R ne pourra passer vers la sortie Y .

  51 Le deuxième verrou est le **maître** et est en mode **transparent**. C'est-à-dire qu'il renvoi directement une sortie Q correspondant à la valeur précédente de Y mémorisée à l'état précédent.

1. Pour un "*Edge-Triggered flip-Flop*" le changement d'état du circuit est déclenché par les signes positifs ou négatifs des flancs. En d'autres termes, si c'est un flanc montant ou descendant.



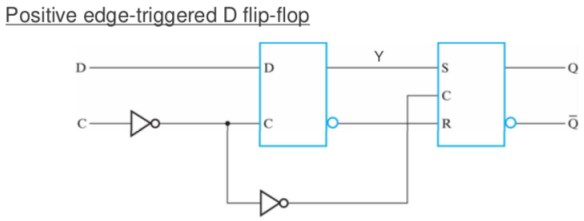
Remarques :

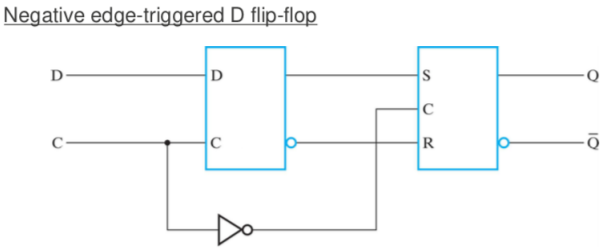
- Dans les deux cas, le chemin reliant les entrées (S , R) et la sortie Q est interrompu.
- Comme pour le verrou SR, le cas où $S = R = 1$ est impossible.

3.7 Question 34

Qu'est-ce qu'un flip-flop déclenché par un flanc de l'horloge (edge-triggered flip-flop) ?

C'est un flip-flop donc le changement d'état est déclenché par le front positif (de 0 à 1) ou négatif (de 1 à 0) de l'horloge.





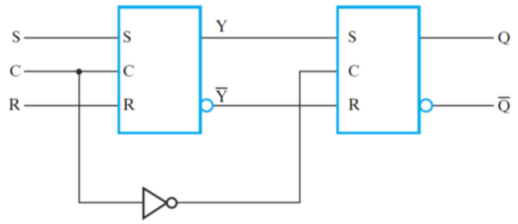
Son fonctionnement s'organise de la sorte :

1. Pour un Positive edge-triggered D flip-flop :
   - Lorsque C = 0, l'entrée est observée depuis le premier latch et transmise à Y tant que C est élevé. Le SR latch est en mode mémorisation et délivre en sortie l'état précédent mémorisé.
   - Lorsque C = 1, le SR latch est en mode transparent. La valeur à Y mémorisée par le latch D à l'instant précédant le passage de 0 à 1 de l'horloge, est transmise à Q.
   - Le changement de la sortie du flio-flop D est associé à la valeur de l'entrée D au front positif de l'impulsion. Observez qu'il n'y a pas de problème de "1st catching.
2. Pour un Negative edge-triggered D flip-flop :
   - Le design est très similaire (signal d'horloge inversé). La valeur qui était à D immédiatement avant que l'horloge tombe de 1 à 0 est copiée sur Q immédiatement après que l'horloge tombe. Dans tous les autres cas, Q conserve son ancienne valeur.

3.8 Question 35

Qu'est ce qu'un flip-flop déclanché par un niveau de l'horloge (pulse-triggered flip-flop) ?

C'est un flip-flop donc le changement d'état est déclenché par la valeur (haute ou basse) de l'horloge. Ceux-ci souffrent généralement de problème de "1st catching".



Son fonctionnement est décrit à la **Question 33**.

3.9 Question 36

Donnez la table de vérité d'un *flip-flop* D

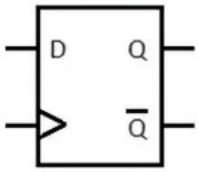


Table of truth:

clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

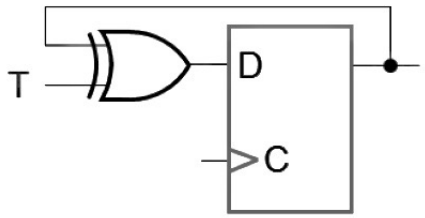
3.10 Question 37

Comment réaliser un *flip-flop* T à partir d'un *flip-flop* D et de portes logiques ?

Le *flip-flop* T fonctionne comme suit :

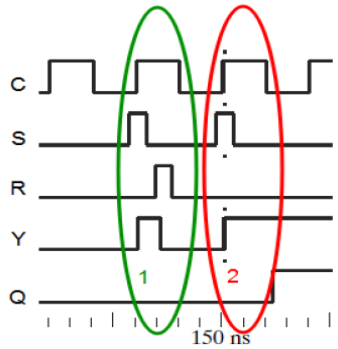
- Si $T = 0$: l'état courant ne change pas $Q(t+1) = Q(t)$.
- Si $T = 1$: l'état est changé en l'état complémenté $Q(t+1) = \overline{Q}(t)$.

On peut facilement réaliser ce *flip-flop* à l'aide d'un *flip-flop* D et d'une porte XOR comme suit :



3.11 Question 38

Expliquez le problème de "1st catching" dans le *flip-flop* SR.



Comme nous le savons, les entrées S et/ou R ne peuvent changer de valeur que lorsque $C = 1$. Plaçons nous dans deux cas bien précis :
Supposons que la sortie Q est initialisée à 0.

- **Cas 1 :**
  L'entrée S passe à la valeur 1 puis reviens à 0. Ensuite, l'autre entrée R passe à la valeur 1 puis reviens à 0, et ce durant tout le temps où l'horloge $C = 1$.
  Dans ce cas ci :

  51 La sortie du maître Y suit la danse et passe à la valeurs 1 puis reviens à 0 lorsque $R = 1$.

  51 la valeur 0 retenue par l'esclave est ainsi passé à la valeur de sortie final Q .

- **Cas 2 :**
  L'entrée S passe à 1 et reviens à 0. Pendant ce temps, R reste à 0.
  Dans ce cas, nous remarquons que :

  51 Y qui est passé par la valeur 1 ne change plus par après.

  51 La valeur 1 est retenue par l'esclave. Du coup, nous avons $Q = 1$.

Un comportement attendu de ce *flip-flop* SR serait que la valeur de la sortie Q corresponde aux valeurs des entrées S et R juste avant le flanc descendant de l'horloge. Dans le premier cas, ce comportement est respecté. Cependant, dans le deuxième cas, comme S et R sont égale à 0 juste avant le flanc descendant de l'horloge, nous devrions avoir une sortie $Q = 0$. Or ici nous avons un $Q = 1$.

3.12 Question 39

Donnez la table de vérité d'un *flip-flop* JK

J	K	$Q(t+1)$	Opération
0	0	$Q(t)$	Pas de changement
0	1	0	Réinitialisation
1	0	1	Initialisation
1	1	$\overline{Q}(t)$	Complémentation

3.13 Question 40

Qu'appelle-t-on état interne (state) d'un circuit séquentiel ?

Un état est défini par une combinaison de valeurs des variables d'état. L'état est lié aux *flip-flops* présents dans le circuit. L'état interne d'un circuit séquentiel fait référence à l'état des *flip-flops* qui composent le dispositif de mémoire.

3.14 Question 41

Qu'est-ce qu'un état indifférent (don't care) ?

Les états indifférents sont des états inutilisé d'un circuits séquentiel qui apparaissent lorsque le nombre d'états disponibles dans le registre d'états ($N = 2^n$, avec n le nombre de flip-flop/verrou) est supérieur au nombre d'états inclus dans le diagramme d'état.

Un état est défini par une combinaison de valeurs des variables d'état. L'état est lié aux *flip-flops* présents dans le circuit. L'état interne d'un circuit séquentiel fait référence à l'état des *flip-flops* qui composent le dispositif de mémoire.

3.15 Question 42

Combien d'états internes peut-on représenter au maximum dans un système séquentiel à N *flip-flops* ?

Pour représenter les états m, nous avons besoin de n bits, avec $n \geq \lceil \log_2 m \rceil$. Un *flip-flop* contient 1 bit, donc, avec n bascules, nous pouvons représenter 2^n états.

3.16 Question 43

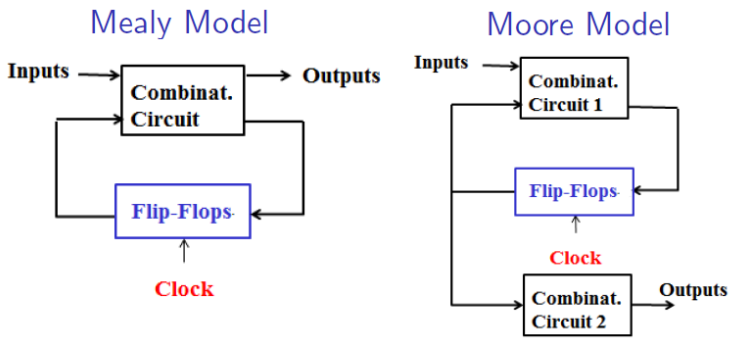
Qu'est-ce qu'une machine d'état de Moore / de Mealy ?

On appelle **machine d'état de Moore** un modèle d'état dont les sorties ne dépendent que de l'état courant. En d'autres termes,

$$\text{output}(t) = f(\text{state}(t))$$

On appelle **machine d'état de Mealy** un modèle d'état dont les sorties dépendent à la fois des entrées et de l'état courant. En d'autres termes,

$$\text{output}(t) = f(\text{state}(t), \text{inputs}(t))$$



3.17 Question 44

Citez différentes manières d'assigner les états d'un système à n variables d'état ? En quoi le choix de la méthode d'assignation peut être importante ?

On peut assigner les états d'un système de différentes manières :

- Par assignation des codes binaires aux états dans l'ordre de comptage et en minimisant le nombre de *flip-flops*.
  Exemple :
  $$A = 00, B = 01, C = 10, D = 11$$
- Par assignation des codes binaires aux états dans l'ordre des codes de Gray et en minimisant le nombre de *flip-flops*.
  Exemple :
  $$A = 00, B = 01, C = 11, D = 10$$
- Par assignation des codes binaires *One-Hot*.
  Exemple :
  $$A = 0001, B = 0010, C = 0100, D = 1000$$

Notons que la dernière méthode permettra d'obtenir des circuits plus simples, mais impliquera la présence d'un plus grand nombre de *flip-flops*. Dans ce cas là également, on remarque qu'on a

$$n > \lceil \log_2 m \rceil$$

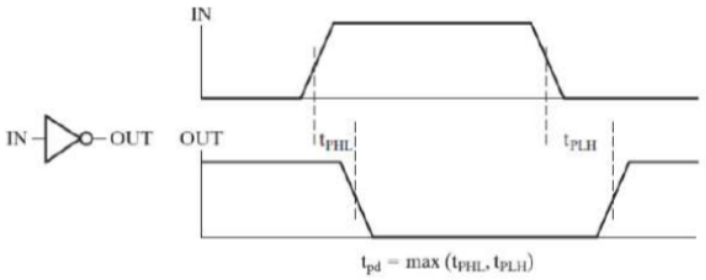
où n est le nombre de bits requis par le circuit et m est le nombre d'états de celui-ci. Dés lors, on aura des cas parasites à traiter lors de la réalisation du système.

3.18 Question 45

Quel est le délai de propagation d'un dispositif numérique ? Quelle différence existe-t-il entre t_{PHL} et t_{PLH} ?

C'est le temps écoulé entre un changement dans une entrée et le changement de sortie qui en découle. t_{PHL} représente le temps de propagation du haut vers le bas (high to low) et t_{PLH} représente le temps de propagation du bas vers le haut (low to high) et graphiquement, on a :

delay usually means at 50 % H and L voltage levels

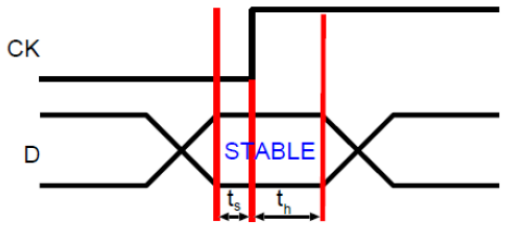


3.19 Question 46

Quels sont les temps caractéristiques à respecter pour assurer le fonctionnement correct d'un flip-flop ?

Il existe **3** temps caractéristiques qui assure le bon fonctionnement d'un flip-flop.

- Le **Setup Time**[2] t_s :
  Il s'agit du temps minimum requis sur lequel les entrées (S ,R ,etc) ne peuvent pas changer AVANT le changement de l'horloge qui déclenche le changement de la sortie.
- Le **Hold Time**[3] t_h :
  Il s'agit du temps minimum requis sur lequel les entrées (S ,R , etc) ne peuvent pas changer APRÈS le changement de l'horloge.
- Le **Propagation Delay**[4] t_{pd} :
  Il s'agit de l'intervalle entre le déclenchement d'un flan de l'horloge et la stabilisation de la sortie sur sa nouvelle valeur.



REMARQUE :

- Pour un "*pulse-triggered flip-flop*" : Le temps t_s est égale à la longueur de l'impulsion (pour palier au problème de "1's catching").
- Pour un "*edge-triggered flip-flop*" : Le temps t_h est plus petit que la longueur de l'impulsion.
- le temps t_h est souvent négligé.
- Le délai de propagation t_{pd} minimum DOIT être plus grand que le temps t_h.

2. Une traduction serait "*Temps d'Installation*" ou "*Temps d'Initialisation*".
3. Une traduction serait "*Temps de retenue*".
4. Une traduction serait "*Temps de propagation*".

3.20 Question 47

Expliquez le *fan-out* d'une porte. Qu'influence-t-il ?

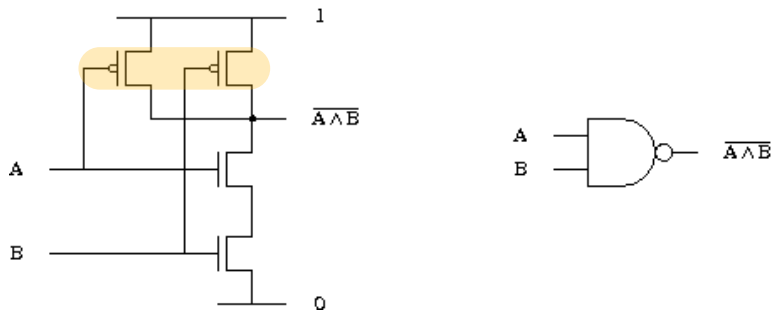
Dans un circuit, on peut dire que chaque entrée d'une porte donnée fournit une charge sur la sortie d'une *porte en marche*[5]. Cette charge est exprimée en terme de "*charge standard*". Dés lors, on obtient le "*fan-out*" en additionnant les contributions de toutes les *portes commandées*[6].

De plus, la détermination de la charge dépend de la technologie utilisée. Pour une technologie CMOS, on dira que la charge égale la capacitance. Cette capacitance a un effet direct sur le temps requis à la sortie d'une porte commandée pour passer d'un voltage haut vers un volage bas OU d'un voltage bas vers un voltage haut. Ce temps requis est appelé "*Temps de transition*[7]". Dés lors, on dira que le "*fan-out*" maximal d'une porte est fixé par le temps de transition maximal.

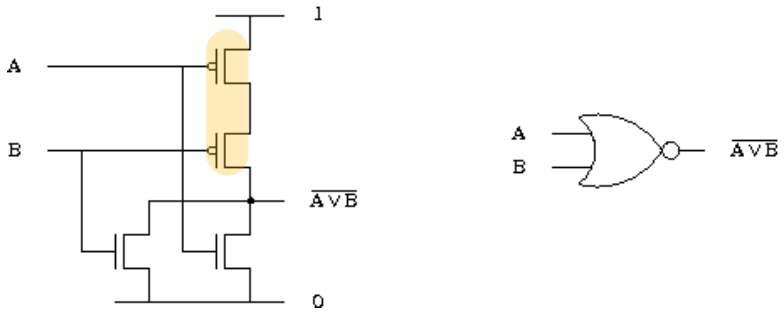
3.21 Question 48

Esquissez le circuit d'une porte NAND/NOR/NOT en utilisant des transistors.

- La porte NAND peut être représenté en logique CMOS par un circuit de 4 transistors (2 n-MOS et 2 p-MOS).

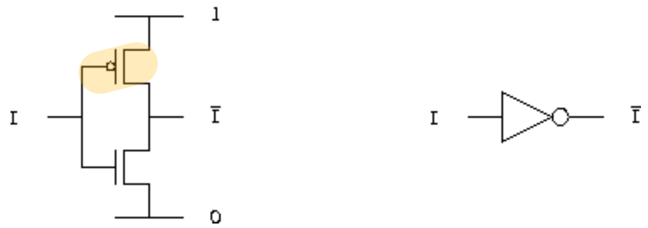


- La porte NOR peut être représenté en logique CMOS par un circuit de 4 transistors (2 n-MOS et 2 p-MOS). C'est le circuit dual du circuit de la porte NAND. Les deux transistors p-MOS qui était en parallèle dans le circuit de la porte NAND sont en série dans le circuit de la porte NOR. Au contraire, les deux transistors n-MOS qui était en série dans le circuit de la porte NAND sont en parallèle dans le circuit de la porte NOR.



5. En anglais : *Driving gate*
6. En anglais : *Driven Gates*
7. En anglais : *Transition Time*

- La porte NOT peut être représenté en logique CMOS par un circuit de 2 transistors.

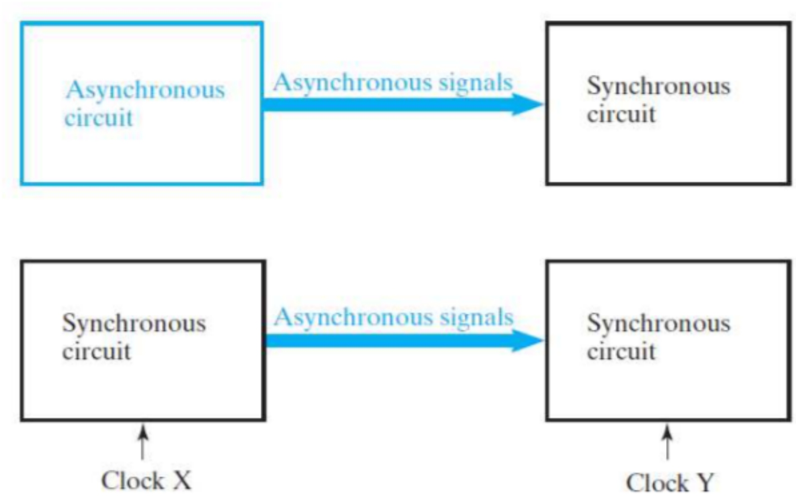


3.22 Question 49

Expliquez l'importance de synchroniser un circuit séquentiel.

Des systèmes synchrone peuvent être sujet à des intéractions asynchrone :

- Lorsqu'ils sont connecté à des systèmes asynchrones.
- Ou lorsqu'ils sont connecté à un système synchrone mais avec une clock différente.



Dans les deux cas, le système est nourri de signals asynchrone dont :

- des changements dans ces signaux asynchrones peuvent arriver aux entrées des flip-flops pendant le temps d'attente d'initialisation
- cela implique que deux flip-flops peuvent réagir différemment, ce qui entraîne un état incorrect.

La synchronisation des signaux d'entrées d'un circuit séquentiel est donc importante.

3.23 Question 50

Qu'est-ce qu'un état métastable ? Comment peut-il survenir ? (pas sur de ma réponse)

---

position entre 0 et 1, et rien pour le faire choisir, c'est le bruit qui peut le faire (...) 1 côté ou de l'autre.

Les flip-flops peuvent entrer dans des états métastables lorsque plusieurs entrées changent simultanément, c'est-à-dire qu'elles peuvent n'offrir une sortie stable qu'au bout d'un temps arbitrairement long. Une entrée asynchrone (circuit combinatoire externe, bouton poussoir), peut causer des métastabilité sur un système synchrone. Afin d'éviter cela, on synchronise cette entrée à l'aide d'un synchroniseur (deux ou plusieurs flip-flop mis en séries).

peut survenir si : when an input to the cross-coupled pair of latch gates changes in just the right timing relationship with the clock edge, or narrow pulse can be generated on line P.291

4 Chapitre 9,10,11

4.1 Question 51

Qu'est ce qu'un registre ?

Un registre à n bits est un ensemble de n éléments de stockage binaires. Plus généralement, un registre est constitué d'un ensemble de n *flip-flops*, éventuellement avec des portes combinatoires supplémentaires. Un registre est utilisé pour :

- Stocker un vecteur de valeurs binaires, appelé mot binaire.
- Effectuer un décalage de mots binaires.
- Effectuer d'autre tâches de traitement de donnée de base, nommées micro-opérations.

4.2 Question 52

Qu'est ce qu'un compteur ?

Un compteur est un circuit séquentiel qui passe par une séquence d'états prescrite lors de l'application d'impulsions d'horloge successives. Un compteur est constitué d'un ensemble de n *flip-flops* interconnectés par des portes combinatoires. Un compteur binaire suit la séquence des nombres binaires, de 0 à $2n - 1$ Les compteurs peuvent aussi compter vers le bas ou compter d'autres séquence fixées de nombres.

4.3 Question 53

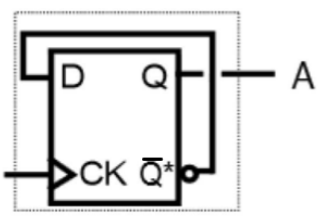
Qu'est ce qu'un *ripple counter* ? Quel est son désavantage ?

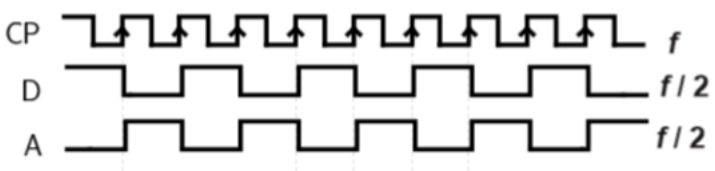
Les compteurs asynchrones (ou *ripple counter*) utilisent les transitions de sortie des *flip-flops* pour déclencher les autres *flip-flops*, ils fonctionnent donc en cascade. Ces compteurs posent quelques problèmes. En effet, ils sont lents, et on peut observer des états transitoires, qui sont causés par les délais de propagation.

4.4 Question 54

Décrivez le fonctionnement du compteur à 1 bit et tracez l'évolution temporelle des variables caractéristiques de ce compteur.

Ce type de compteur n'utilise qu'un seul *flip-flop*. Quand le signal de *clock* est positif, la sortie A est complémentée. Ce type de compteur est aussi appelé un diviseur de fréquence, car la fréquence de la sortie Q est égale à la moitié de la fréquence du signal de *clock* C. L'évolution temporelle des variables caractéristiques de ce compteur est donnée par :





4.5 Question 55

Qu'est-ce qu'un compteur synchrone/(asynchrone) ?

- Un compteur synchrone utilise une *clock* commune pour tout ses *flip-flops*.
- (Un compteur asynchrone (ou *ripple counter*) utilise les transitions de sortie des *flip-flops* pour déclencher les autres *flip-flops*.)

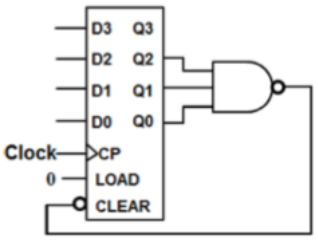
4.6 Question 56

Comment initialiser un compteur ? Donnez un exemple de bonne pratique et un exemple de mauvaise pratique.

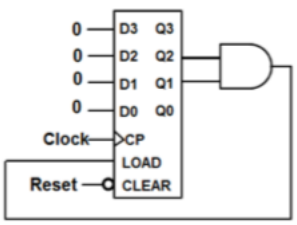
Un compteur doit être initialisé et de façon synchrone. On peut utiliser un compteur général, en utilisant l'entrée `Load`. Il ne faut surtout pas utiliser de *flip-flops* asynchrones. L'initialisation doit être faite seulement lorsque l'on met le compteur sous tensions ou lors d'un *reset*.

Imaginons que nous voulions fabriquer un compteur modulo-7 en utilisant un compteur 4 bit, différentes manières de procéder se présentent alors.

Une solution intuitive serait de détecter le dernier état 0111 et puis utiliser l'entrée *clear* du compteur pour revenir à l'état initial 0000. Le problème est que l'entrée *clear* agit directement sur les entrées *resets* (asynchrones) des *flip-flops*. Ainsi, la transition vers l'état 0000 ne se déroulera pas l'*edge* de la *clock*. L'existence de l'état 0111 ne sera peut-être donc pas assez longue pour permettre de remettre tous les *flip-flops* à 0. Ceci est implémenté de la manière suivante :



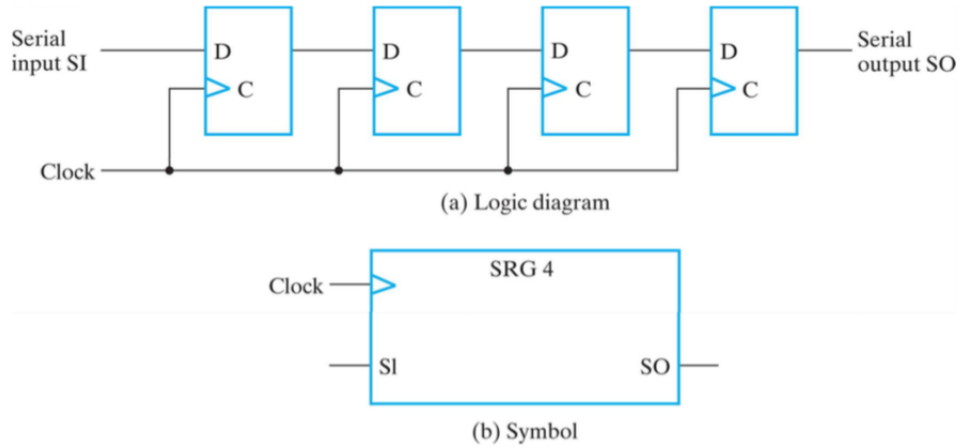
Une meilleure solution consisterait alors à détecter l'état 6, c'est à dire 0110. A ce moment, Q_2 et Q_1 sont égaux à 1, et ceci donne la valeur 1 à *Load*. On charge alors des 0 sur les entrées D_1 à D_3, et lors du prochain coup de *clock*, le compteur va changer son état vers 0000 de manière synchrone. Un tel circuit est représenté ainsi :



4.7 Question 57

Qu'est-ce qu'un registre à décalage ? Donnez des façons séquentielles et combinatoire pour implémenter un tel registre.

Un registre à décalage est constitué d'une chaîne de flip-flop partageant tous le même signal d'horloge. Il permet de déplacer les données dans le registre, que ce soit vers la position la plus ou la moins significative.



4.8 Question 58

Donnez des applications d'un registre

Les registres ont plusieurs applications. Ils permettent :

- Le stockage temporaire d'information
- La conversion série-parallèle et parallèle-série de mots binaires.
- Le combinaison des deux types de conversions cité ci-dessus afin d'interfacer des systèmes numériques éloignés les uns des autres.
- La division par 2^n en effectuant n opérations de décalage vers la droite.
- La multiplication par 2^n en effectuant n opérations de décalage vers la gauche.

4.9 Question 59

Qu'est-ce que le *datapath* dans une description RTL ?

Dans un système digital, il existe 2 types d'informations, les données et les informations de contrôles. Le *datapath* est la partie du système effectuant le traitement des données. Cette partie du circuit transmet également les données à l'unité de contrôle tout comme l'unité de contrôle fournit des donnés au *datapath* sur les opérations devant être réalisées.

4.10 Question 60

Qu'appelle-t-on micro-opération dans une description RTL ?

Une micro-opération est une opération élémentaire agissant identiquement sur chaque bit du circuit et durant au plus un cycle d'horloge.

4.11 Question 61

Quelles sont les 4 catégories de micro-opérations ?

On peut les classer en 4 catégories :

- De transfert : elle copie de l'information d'un registre à un autre.
- Arithmétique : effectue des opérations arithmétique sur des données au sein de registres.
- Logique : manipule des données ou effectue des opération bit à bit.
- De décalage : décale des données dans le registre.

4.12 Question 62

Expliquez la micro-opération "transfert conditionnel entre registres".

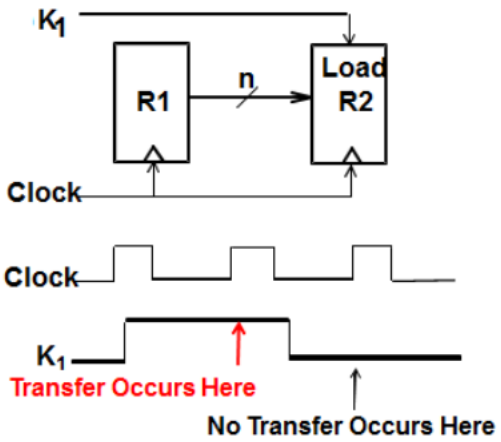
Un transfert conditionnel de donnée entre registre est une micro-opération s'effectuant lorsqu'une certaine condition est satisfaite. On peut écrire :

$$\text{Si } (K_1 = 1) \text{ alors } (R2 \leftarrow R1)$$

ce que l'on note encore :

$$K_1 : (R2 \leftarrow R1)$$

où K_1 est la variable de contrôle agissant comme valeur *Load* du registre $R2$. En-effet, on sait alors que dés que cette conditions sera respectée, le registre $R2$ en série avec $R1$ chargera l'information contenue dans ce dernier.



4.13 Question 63

Donnez au moins 4 exemples de micro-opérations, dont 2 conditionnelles.

- **L'opération d'addition/soustraction arithmétique :** les opérations d'addition et de soustraction sont effectuées dans un circuit commun en incluant une porte XOR exclusif avec chaque additionneur complet. Elles sont données par les instructions : $R3 \leftarrow R1+R2+1, R3 \leftarrow R1+R2$.
- **La sélection sélective :** lors de la sélection sélective, le modèle de bits dans B est utilisé pour définir certains bits dans A. Prenons pour exemple les registres $1100 = A_{(t)}$, $1010 = B$ et $1110 = A_{t+1}$. On a donc l'opération : $A \leftarrow (A + B)$. Si un bit dans B est mis à 1, cette même position dans A est mise à 1, sinon ce bit garde sa valeur précédente dans A. La micro-opération OR peut être utilisée pour définir sélectivement des bits d'un registre.
- **L'opération *clear* :** dans ce type d'opération, si les bits à la même position dans A et B sont identiques, ils sont effacés de A, sinon ils restent défini de la même manière dans A. Par exemple, $1100 = A_{(t)}$, $1010 = B$ et $0110 = A_{(t+1)}$, ce qui défini l'opération $(A \leftarrow A \oplus B)$. Cet opération compare les mots dans A et B et produit tous un résultat rempli de 0 si les deux nombres sont égaux. Cette opération peut être réalisée par la micro-opération XOR.
- **Le masquage :** c'est une opération de logique utilisée pour sélectionner, dans un groupe de bits, un sous-ensemble de bits à conserver, ou au contraire à écraser. Illustrons cette opération à l'aide des registres de 4 bits suivants : $1100 = A_{(t)}$, $1010 = B$ et $1000 = A_{(t+1)}$. On a donc l'opération $(A \leftarrow A.B)$. Le masquage fonctionne comme suit : si un bit dans B est mis à 0, la même position dans A est mise à 0, sinon elle reste inchangée.

4.14 Question 64

Qu'appelle-t-on BUS dans une description RTL ?.

Les systèmes digitaux possèdent beaucoup de registres. Il est nécessaire de définir des canaux de communication entre des différents registres pour transférer des données d'un registre vers un autre. Une solution à ce problème consiste à utiliser un bus, c'est à dire un canal de communication commun à tous les registres, et d'ensuite utiliser un multiplexeur pour sélectionner la source. On peut implémenter les bus en utilisant :

- Des multiplexeurs.
- Un buffer à 3 états.

4.15 Question 65

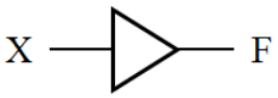
Qu'est ce qu'un circuit *buffer* ? Quelle est son utilité ?

Un *buffer* produit la fonction logique ZX, où la valeur binaire de sortie est égale à la valeur binaire d'entrée. Ce circuit est notamment utilisé pour amplifier un signal électrique. Ceci permet alors d'avoir plus de portes attachés à la sortie, et de diminuer le temps de propagation des signaux dans le circuit. Il permet aussi d'éviter la perte d'information durant sa propagation au sain d'un circuit.

4.16 Question 66

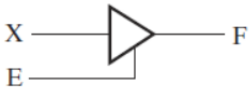
Qu'est-ce qu'un *buffer* 3 états

Rappelons qu'un *buffer* classique est défini à la question 8, c'est un simple composant recopiant une variable :



X	F
0	0
1	1

Un *buffer* 3 états quant à lui permet de renvoyer 3 états comme son nom l'indique, à savoir $[0; 1; \text{High-Z}]$. L'état High-Z représente un état pour lequel le circuit est ouvert. La table de vérité de ce composant est donnée par :



E	X	F
0	0	Hi-Z
0	1	Hi-Z
1	0	0
1	1	1

Ce élément possède une entrée de données Z ainsi qu'une entrée de contrôle E. On voit que si E est à 0, aucune information n'est communiquée par la sortie de cette porte logique.

5 Chapitres 12

5.1 Question 67

Que signifie les acronymes RAM/ROM/EEPROM ?

- RAM : **R**andom **A**ccess **M**emory, c'est la mémoire vive (ne garde pas l'information hors tension).
- ROM : **R**ead-**O**nly **M**emory, c'est la mémoire morte (garde l'information même hors tension). Son contenu est défini lors de la fabrication.
- EEPROM : **E**lectrically-**E**rasable **P**rogrammable **R**ead-**O**nly **M**emory, c'est un type particulier de mémoire morte dont le contenu peut être facilement effacé à l'aide d'un courant électrique.

5.2 Question 66

Qu'appelle-t-on mémoire dynamique/statique ? Quelles sont leur principales différences ?

La mémoire vive statique (ou **SRAM** pour l'anglais **S**tatic **R**andom **A**ccess **M**emory) est un type de mémoire utilisant des verrous et/ou des *flip-flops* afin de mémoriser des données. La mémoire vive dynamique (ou **DRAM** pour l'anglais **D**ynamic **R**andom **A**ccess **M**emory)) quant à elle est un type de mémoire composée de cellules elles même composées d'un transistor et d'un condensateur. La grosse différence entre ces deux mémoires est que la seconde a besoin d'un rafraîchissement permanent de la charge du condensateur sinon l'information contenue dans la cellule DRAM est perdue. Cependant, ce type de mémoire est compacte et moins cher que la première, qui n'exige pas de rafraîchissement puisque des bascules la compose. On peut esquisser ces 2 types de mémoires comme suit :

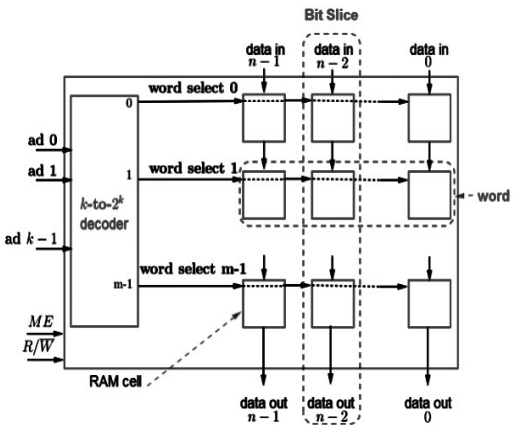


FIGURE 1 – SRAM.

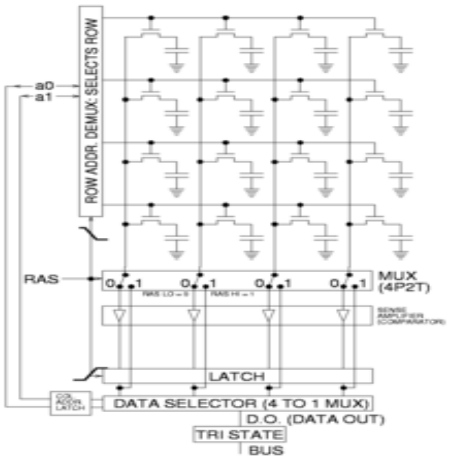


FIGURE 2 – DRAM.

❑ Comparison between SRAM and DRAM:

	SRAM	DRAM
Cost	high	moderate
Speed	high	slower
Refresh	No	Yes
Power consumption	high	low

FIGURE 3 – Tableau comparatif.

5.3 Question 69

Qu'appelle-t-on « rafraîchissement » d'une mémoire dynamique ?

C'est un processus qui consiste à lire périodiquement les données d'une mémoire et à les réécrire immédiatement sans aucune modification afin de ne pas les perdre.

5.4 Question 70

Qu'appelle-t-on mémoire volatile ?

Si l'information stockée dans une mémoire est perdue lors de sa mise hors tension, on dit de la mémoire qu'elle est volatile. Ce type de mémoire a donc besoin d'une source de tension extérieure pour conserver des données.

5.5 Question 71

Qu'est-ce qu'un mot (word) et une adresse (address) dans la description d'une mémoire ?

Un mot est un ensemble d'octets (8 bits) dont la taille est l'unité typique d'accès à la mémoire. C'est également l'entité de bits qui sort et rentre de celle-ci. Chaque mot est représenté par son adresse, étant un nombre binaire qui permet de l'identifier dans la mémoire. Pour m mots, $\lceil k = \log_2 m \rceil$ adresses sont nécessaires.

6 Questions longues

6.1 Question 1

Esquissez le schéma d'un circuit ou d'une opération logique spécifié(e) (par ex. : un décodeur, encodeur, multiplexeur, compteur, "3-bit ripple carry adder", "multiplexeur-based conditional register transfer",...).

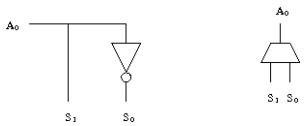


FIGURE 4 – Schéma et symbole d'un décodeur 1 bit

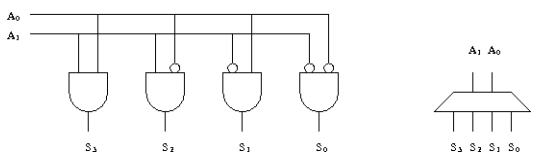
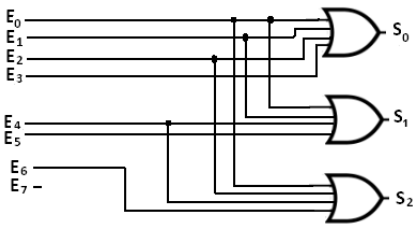


FIGURE 5 – Schéma et symbole d'un décodeur 2 bit



Décimal	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0	0	1
2	0	0	0	0	0	1	0	0	0	1	0
3	0	0	0	0	1	0	0	0	0	1	1
4	0	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	0	0	0	0	1	0	1
6	0	1	0	0	0	0	0	0	1	1	0
7	1	0	0	0	0	0	0	0	1	1	1

FIGURE 6 – Schéma d'un encodeur à 8 entrées

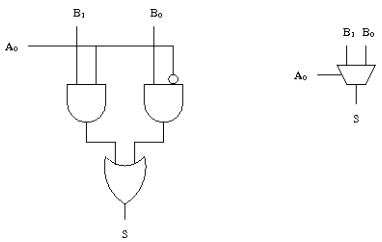


FIGURE 7 – Schéma et symbole d'un multiplexeur 1 bit

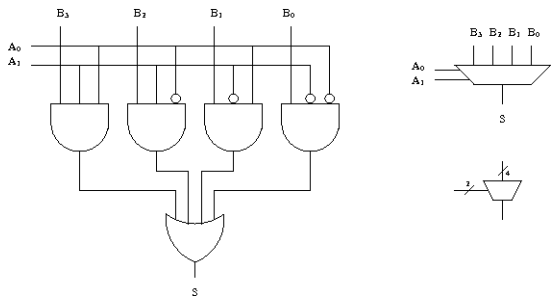


FIGURE 8 – Schéma et symbole d'un multiplexeur 2 bit

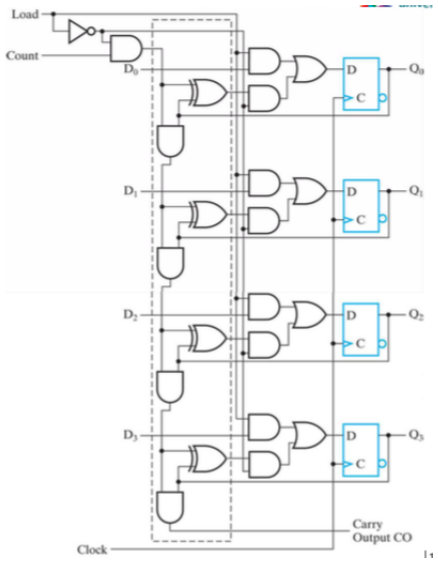


FIGURE 9 – Schéma d'un compteur 4 bit

6.2 Question 2

Énoncer un maximum de manières d'implémenter une fonction combinatoire en indiquant quand il est intéressant de procéder à une simplification de la fonction.

On peut implémenter une fonction de différentes façons. La première consiste à utiliser les fonctions logiques de base suivante :

- AND : XY
- OR : $X + Y$
- NOT : $\overline{X}$

On peut ensuite utiliser différentes identités pour simplifier notre fonction :

1 variable

1. $X + 0 = X$
2. $X . 1 = X$
3. $X + 1 = 1$
4. $X . 0 = 0$
5. $X + X = X$
6. $X . X = X$
7. $X + \overline{X} = 1$
8. $X . \overline{X} = 0$
9. $\overline{\overline{X}} = X$

Rem : $\overline{X}$=X'=~X

Plusieurs variables

10. $X + Y = Y + X$
11. $X + (Y + Z) = X + Y + Z$
12. $X . (Y + Z) = X . Y + X . Z$
13. $X . Y = Y . X$
14. $X . (Y . Z) = X . Y . Z$
15. $X + (Y . Z) = (X + Y) . (X + Z)$
    $\rightarrow X + (X . Y) = X . (X + Y) = X$

Théorème De Morgan

$$\overline{X_1 + X_2 + \ldots + X_n} = \overline{X_1} . \overline{X_2} . \ldots . \overline{X_n}$$
$$\overline{X_1 . X_2 . \ldots . X_n} = \overline{X_1} + \overline{X_2} + \ldots + \overline{X_n}$$

Théorème Du Consensus

$$X.Y + \overline{X}.Z + Y.Z = X.Y + \overline{X}.Z$$
$$(X + Y) . (\overline{X} + Z) . (Y + Z) = (X + Y) . (\overline{X} + Z)$$

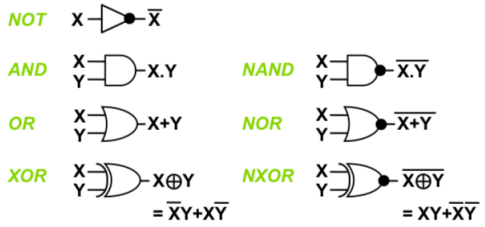
La meilleure façon de simplifier la fonction est d'utiliser les tables de Karnaugh.

Toutes ces méthodes de simplification sont très utiles, elles permettent notamment de minimiser le nombre de fonctions de niveaux et donc le délai total, ainsi que le nombre de portes et donc la taille du circuit combinatoire de la fonction.

6.3 Question 3

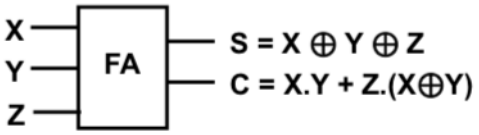
Donner plusieurs méthodes d'implémentation matérielle d'une fonction combinatoire.

On peut implémenter une fonction combinatoire de différentes manières. Il est bien évidemment possible d'utiliser les différentes portes logiques suivantes :

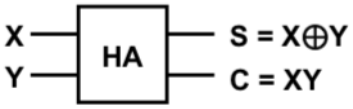


On peut aussi utiliser différents composants logiques, comme les Half-Adder, les Full-Adder, les Multiplexeurs et les Décodeurs.

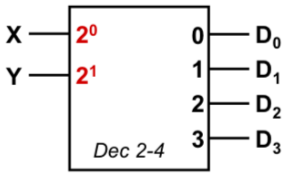
Full-Adder



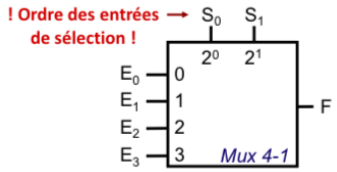
Half-Adder



Décodeur



Multiplexeur



Enfin, l'utilisation des portes NAND et NOR permet de diminuer le nombre de transistors du circuit.

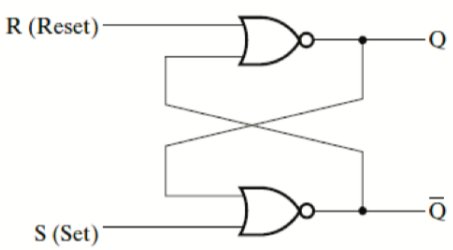
Up plutôt moins de transistors car porte universelle

6.4 Question 4

SR LATCH :

- Tracez le circuit d'un SR Latch en utilisant des portes NOR.
- Expliquez son fonctionnement et donnez sa table d'états
- Illustrez le fonctionnement du verrou par un diagramme temporel
- Quand et comment peut-on observer une "race condition" ?

Un circuit SR Latch peut facilement être représenté avec des portes NOR. Il prend la forme suivante :



Le circuit possède deux états utiles :

- L'état Set : $Q = 1$, $\bar{Q} = 0$
- L'état Reset : $Q = 0$, $\bar{Q} = 1$

Il possède deux entrées :

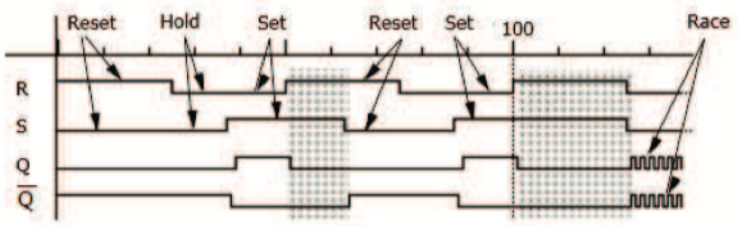
- L'entrée Set S : quand $S = 1$, le circuit passe à l'état Set
- L'entrée Reset R : quand $R = 1$, le circuit passe à l'état Set

Un SR latch se comporte donc de la manière suivante :

1. Le système démarre avec R et S tous les deux égaux à 0. L'état stocké est donc initialement inconnu.
2. S passe alors à 1, ce qui initialise alors Q à 1.
3. S repasse à 0, mais Q retient la valeur 1, le système est à l'état Set.
4. R passe à 1, et Q est donc réinitialiser à 0.
5. R repasse à 0, mais Q retient la valeur 0, le système est à l'état Reset.
6. Si l'on suppose que R et S passent tous les deux à 1, alors Q et $\bar{Q}$ valent tous deux zéro. Nous sommes alors d'un état indéfini.
7. Si R et S passent tous les deux à zéro simultanément, l'état du circuit sera instable et oscillera entre 00 et 11, deux états non définis. C'est dans ce cas-là qu'on obeservera alors la "race condition" (point 4 de la question)

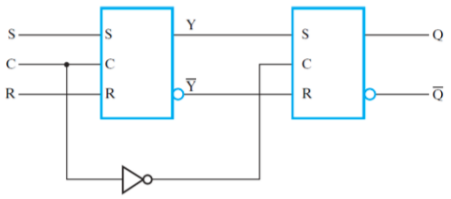
Ceci peut être représenté grâce à la table d'état et au diagramme temporel suivant représentés ci-dessous.

Time	R	S	Q	$\bar{Q}$
	0	0	?	?
	0	1	1	0
	0	0	1	0
	1	0	0	1
	0	0	0	1
	1	1	0	0
	0	0	?	?



6.5 Question 5

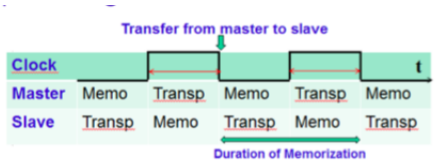
Sur base du circuit donné ci-dessous du « SR master-slave flip-flop », définissez et explicitez les deux modes de fonctionnement observés selon la valeur de la variable de contrôle C.

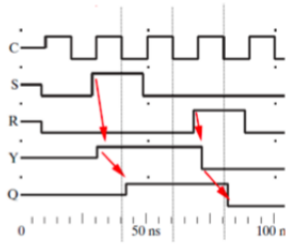


Le « SR master-slave flip-flop » est composée de 2 SR Latch connectés en cascade. Le signal de clock est inversé pour le 2ème latch. En fonction de la variable de contrôle C, on observe deux comportements différents du circuit .

- Lorsque $C = 1$, le premier latch est considéré comme le maître et il est alors en mode transparent. Le second latch est l'esclave et il en mode mémorisation. L'entrée du circuit est donc passé à la sortie Y, mais il ne passe pas à la sortie Q.
- Lorsque $C = 0$, le premier latch est considéré est en mode mémorisation, alors que me second latch est lui en mode transparent. La sortie Y ne changera donc pas, même si les variables d'entrées S et R sont modifiées, et la valeur de Y mémorisée par le premier latch et passée à Q

Dans les deux cas, les variables S et R ne peuvent pas directement accéder à Q, ce qui permet un comportement plus stable. Malgré tout, il est interdit de choisir $S = R = 1$. On peut donc représenter le comportement temporel du SR master-slave flip-flop ainsi :





6.6 Question 6

Concevez le circuit combinatoire à ajouter à un flip-flop specifié* pour le transformer en flip-flop spécifié*.

A voir à l'examen

6.7 Question 7

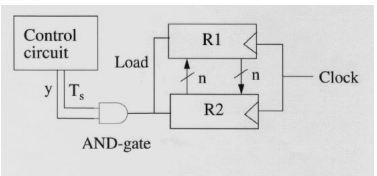
Dessinez le schéma d'un compteur spécifié* construit avec des flip-flops spécifiés* et une entrée enable.

A voir à l'examen

6.8 Question 8

Tracez le schéma qui implémente la micro-opération suivante en RTL

$$C3 : R2 \leftarrow R1, R1 \leftarrow R2$$

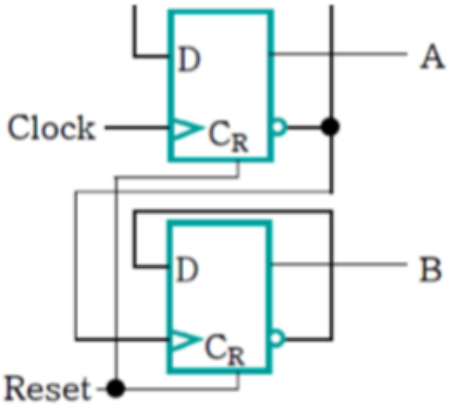


Cette opération simultanée peut être effectuée à l'aide de registres ayant des bascules déclenchées par les flancs. Les registres sont connectés les uns aux autres et le signal de la porte ET sert de signal *load* (charge).

6.9 Question 9

Tracez le circuit d'un compteur asynchrone (ripple counter). Décrivez son fonctionnement et montrez qu'il existe des états transitoires induits par les délais de propagation des flips-flops.

Les compteurs asynchrones (ou *ripple counter*) utilisent les transitions de sortie des *flip-flops* pour déclencher les autres *flip-flops*, ils fonctionnent donc en cascade. Pour un compteur a 2 bits, on a le schéma suivant :



Symbol

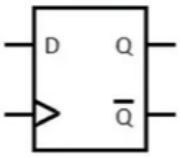
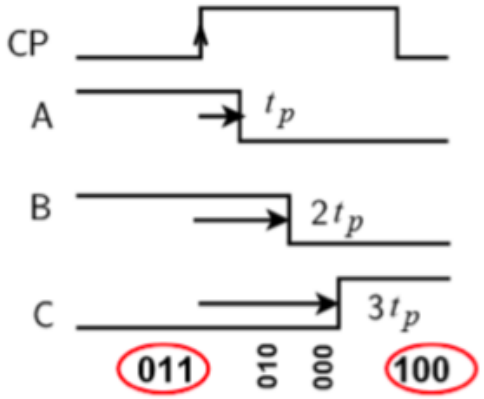


Table of truth:

clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

On remarque sur le schéma que la sortie A du premier flip-flop est complémentée lors d'une rising edge de la clock de ce flip-flop. Cette même sortie A est la clock du second flip-flop. Ainsi, lorsque A passe de 0 à 1, la sortie B du second flip-flop est complémentée. Le compteur passe donc sur la séquence d'états suivante : $00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00$.

Le délai de propagation de chaque flip-flop est de t_p, pour n flips-flops, le délait est donc de nt_p. On observe aussi des états sur ces compteurs. En effet, pour un compteur modulo-8 par exemple, le compteur va passer au travers des états transitoire 010 et 000 avant d'atteindre l'état 100. Ceci est du aux délais de propagations des flips-flops. Ceci est représenté sur le schéma temporel suivant :



6.10 Question 10

Soit le diagramme d'états specifié*, déterminez la table d'états correspondante.

Exemple de cas :

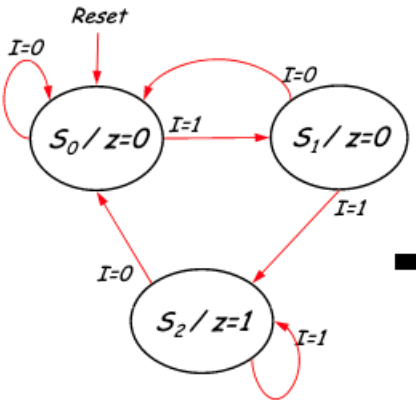


Diagramme d'état pour FSM de Moore

La sortie z dépend uniquement de l'état courant

État présent	I	État suivant	z
S_0	0	S_0	0
S_0	1	S_1	0
S_1	0	S_0	0
S_1	1	S_2	0
S_2	0	S_0	1
S_2	1	S_2	1

Table d'état

6.11 Question 11

Soit la table d'états spécifiée*, déterminez le diagramme d'états correspondant.

Exemple de cas :

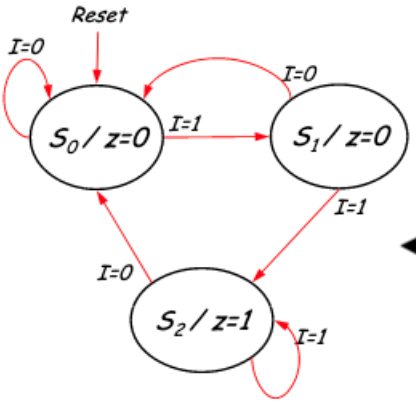


Diagramme d'état pour FSM de Moore

La sortie z dépend uniquement de l'état courant

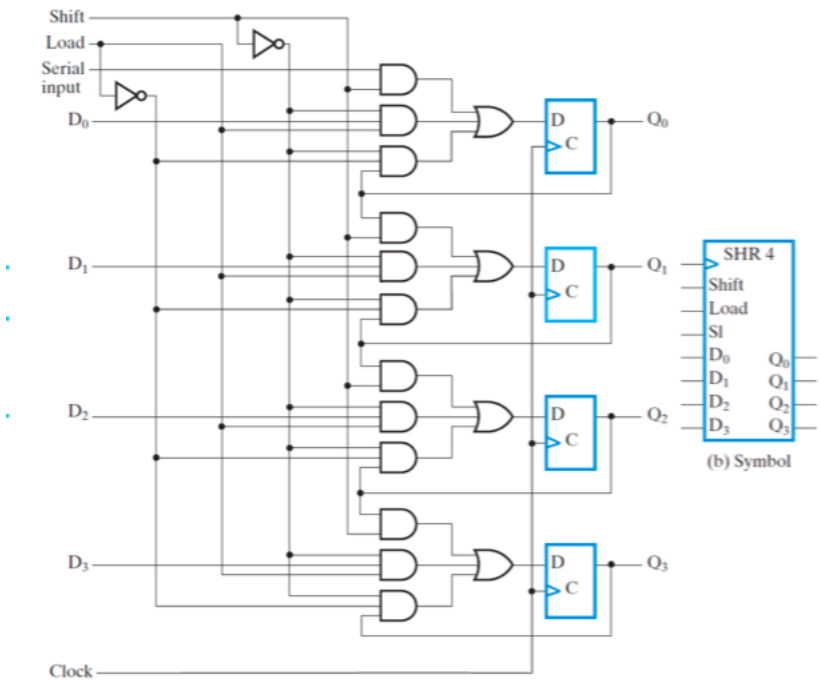
État présent	I	État suivant	z
S_0	0	S_0	0
S_0	1	S_1	0
S_1	0	S_0	0
S_1	1	S_2	0
S_2	0	S_0	1
S_2	1	S_2	1

Table d'état

6.12 Question 12

Tracez le circuit d'un registre à décalage 4-bits avec chargement parallèle et mode "maintien" (hold on) et expliquez son fonctionnement.

Ce type de registre est représenté ainsi :

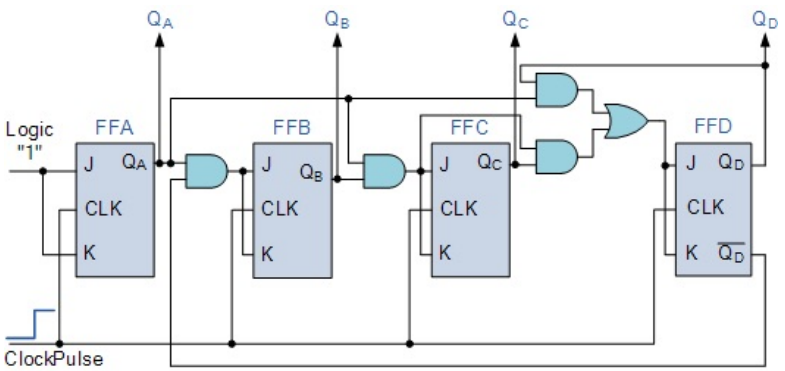


Il fonctionne comme un registre avec chargement parallèle (voir son fonctionnement à la question 55) normal mais possède un mode en plus. Ce mode, appelé le mode maintien est ajouté en ajoutant des portes AND.

6.13 Question 13

Esquissez le circuit d'un compteur BCD, donnez sa table d'états ainsi que ses équations d'états.

Le circuit peut être représenté comme suit :



La table d'état est alors donnée par :

Present state				Next state			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0

On obtient alors les équations suivantes :

$$Q_8(t+1) = Q_8\bar{Q_1} + Q_1Q_2Q_4$$
$$Q_4(t+1) = Q_4 \oplus Q_1Q_2$$
$$Q_2(t+1) = Q_2\bar{Q_1} + \bar{Q_8}\bar{Q_2}Q_1$$
$$Q_1(t+1) = \bar{Q_1}$$

6.14 Question 14

A partir du schéma spécifié* d'une mémoire, décrivez le principe d'adressage d'un mot.

A voir à l'examen

6.15 Question 15

Considérons une mémoire $m \times n$ (m et n sont spécifiés*)

- Quelle est la dimension d'un décoder de lignes et de colonnes ?
- Pour un mot stocké à l'adresse spécifiée*, déterminez la sortie sélectionnée par le décodeur ligne et la sortie sélectionnée par le décodeur colonne.

Exemple de cas :

- ❑ The same RAM cell array can be used to produce a 8 x 2 RAM chip.
- ❑ 8 words require 3 address bits:
  - Row decoder: 2-to-4.
  - Column decoder : 1-to-2.
- ❑ There are two *Data In* inputs, one for each bit of data.
- ❑ Each *Column Select* selects two columns, one for each *Data In* bit.
- ❑ Example:
  - address 001 corresponds to cells 2, 3.
  - two *Data Out* outputs.

6.16 Question 16

Quelles sont les caractéristiques (nombres de bits, synchrone/asynchrone, série/parallèle, modes opératoires, rôles des variables de contrôle) du compteur ou additionneur représenté sur le schéma *donné**.

A voir à l'examen

6.17 Question 17

Expliquez et esquissez une cellule SRAM / DRAM (en utilisant un latch / en utilisant des transistors).

---

1. L'unité de stockage de base d'une mémoire SRAM est composée de 6 transistors pouvant être implémentés de manière logique par un verrou SR.

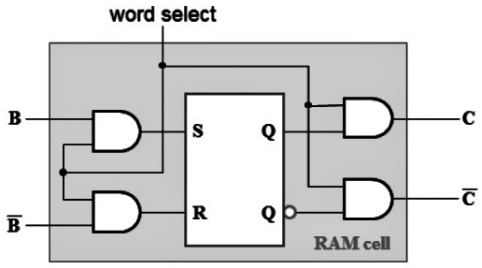


FIGURE 10 – SRAM logique.

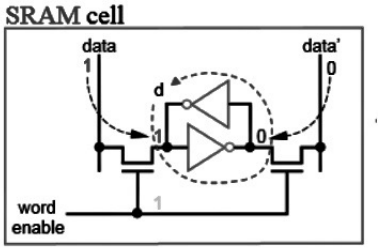


FIGURE 11 – SRAM électrique.

Son comportement est le suivant :

- Si *word select* $= 0 : C = \bar{C} = 0$ et la sortie Q retient la la valeur stockée.
- Si *word select* $= 1 : C = B, \bar{C} = \bar{B}$ et la valeur est mise à jour.

C'est donc *word select* qui signale si on réalise une opération d'écriture ou de lecture dans la mémoire. B et $\bar{B}$ sont utilisé lors de l'écriture alors que C et $\bar{C}$ le sont lors de la lecture.

2. L'unité de stockage de base d'une mémoire DRAM est composée d'un transistor et d'un condensateur pouvant être implémentés de manière logique par un verrou D et une porte *buffer* 3 entrée.

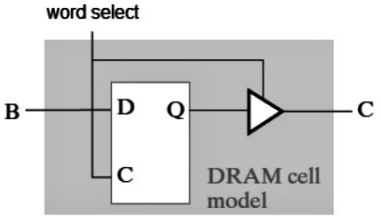


FIGURE 12 – DRAM logique.

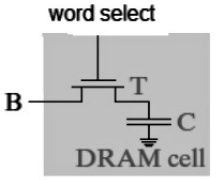


FIGURE 13 – DRAM électrique.

Du point de vue électrique, le condensateur stock une charge électrique (haute pour 1 et basse pour 0). Le transistor quant à lui agit comme un interrupteur, si celui-ci est ouvert la valeur contenue dans la cellule est retenue, si il est fermé une nouvelle valeur y est transférée. Ce circuit

doit être en permanence rafraîchit puisque le condensateur perd de sa charge avec le temps, dés lors il existe dans circuit effectuant ce rafraîchissement lorsque le transistor est ouvert pour ne pas perdre l'information. Durant une opération de lecture, la charge du condensateur varie, il faudra donc recopier immédiatement l'information pour en éviter toute perte.

Du point de vue logique, *word select* informe sur la nature de l'opération sur la mémoire (lecture ou écriture). Le fonctionnement reste le même que pour la mémoire SRAP si ce n'est qu'un porte tampon à 3 accès y est utilisé ainsi qu'un amplificateur pour adapter les variations de tensions en niveau haut et bas.