Labo n° 5
Électronique appliquée [ELEC-H-301]
Circuits logiques
Corrigé

2016–2017

1 Introduction

1.1 But

Cette manipulation a pour but d'illustrer :
- au niveau «application» : le fonctionnement de circuits logiques câblés
- au niveau «composant» : le fonctionnement de circuits logiques discrets.

1.2 Prérequis

Chapitres n° 24 et n° 25 du livre de référence (ed 5).
En particulier :
- algèbre de Boole
- portes logiques, états logiques
- logique combinatoire
- délai de propagation
- logique séquentielle : bascules R-S et D.

Veillez aussi à relire l'annexe du laboratoire n° 1 sur les instruments de laboratoire.

1.3 Prédéterminations

Les questions 1, 2, 5, 7 à 13, 15 et 16 doivent être faites **avant** l'arrivée au laboratoire.

1.4 Objectifs

À la fin de ce laboratoire, vous devez être capable de :
- donner la table de vérité d'un circuit logique
- réaliser et tester un circuit logique
- comprendre la notion de temps de propagation et le mesurer à l'oscilloscope
- comprendre le fonctionnement d'une bascule R-S
- réaliser et tester une bascule D à l'aide de portes logiques.

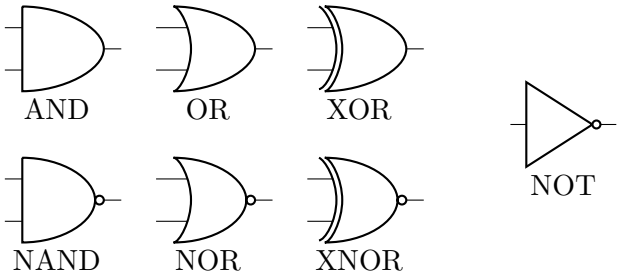
2 Concepts

Les circuits logiques sont invisibles mais fortement présent autour de nous : ordinateurs, téléphones, tablettes numériques... Même si la complexité des circuits a fortement augmenté depuis quelques décennies, les fonctions de base sont restées essentiellement identiques. Ce labo a pour but de vous montrer comment réaliser les fonctions de base de la logique numérique avec des circuits conçus à cet effet.

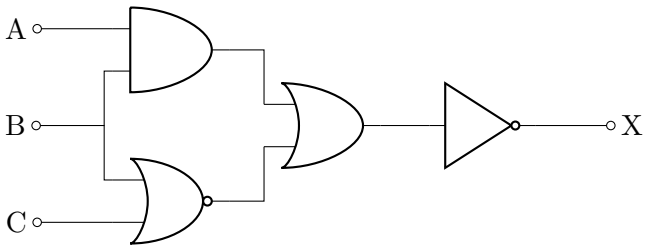
3 Logique combinatoire

Les circuits décrits dans cette section sont **combinatoires**, c'est à dire que leur sortie ne dépend que de l'état des entrées.

Les opérations logiques usuelles [1] sont :



Question 1. Donner la table de vérité du circuit suivant :

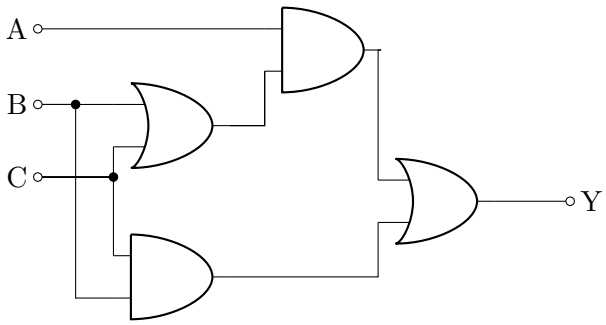


Réponse :

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

1. La représentation américaine des portes logiques est utilisée ici, une représentation européenne existe également

Question 2. En utilisant une table de vérité, démontrer que ce circuit est un «voteur» :



Réponse :

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

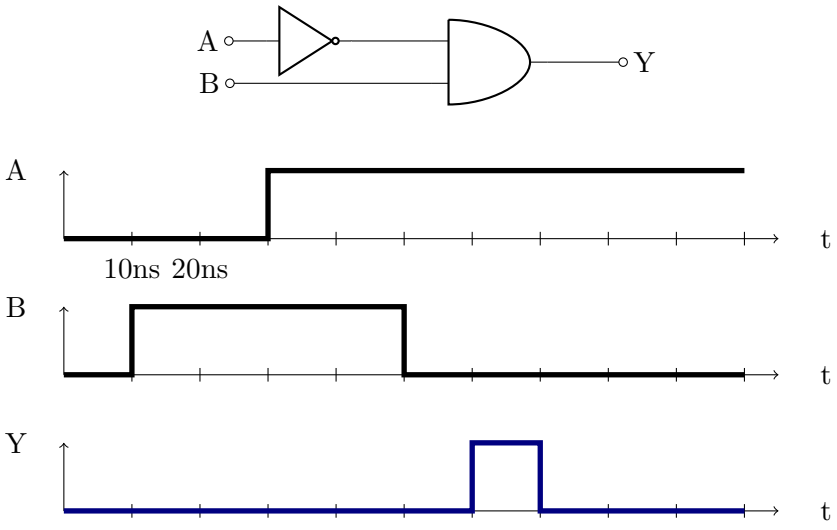
Question 3. Réaliser ce circuit sur le protoboard au moyen de circuits intégrés 74HC00 (**N**AND) et 74HC32 (OR), les brochages se trouvent en annexe.

- Brancher les entrées sur les leds pour visualiser leur état,
- brancher les sorties intermédiaires sur les leds pour vérifier progressivement votre câblage,
- brancher la sortie sur une led pour vérifier son état.
- **N'oubliez pas d'alimenter vos circuits logiques en 5V !**

Note : Réalisez d'abord un circuit permettant de simuler le comportement d'une porte AND à l'aide de deux portes NAND.

Question 4. Tester son bon fonctionnement à l'aide des LEDs et des interrupteurs. Vérifier si sa table de vérité correspond à celle déterminée précédemment.

Question 5. Compléter le chronogramme ci-dessous en considérant que la porte "NOT" a un temps de propagation de 10ns et que la porte "AND" a un temps de propagation de 20 ns.



Question 6. Mettre en évidence le temps de propagation sur le circuit *de la question 2.*

- Brancher A sur l'entrée de synchronisation de l'oscilloscope et l'utiliser pour le déclenchement de l'oscilloscope,
- brancher la sortie de $A \cdot (B + C)$ sur `CH1`,
- brancher Y sur `CH2`,
- utiliser les curseurs et le retard par rapport au déclenchement pour déterminer les temps de propagation.

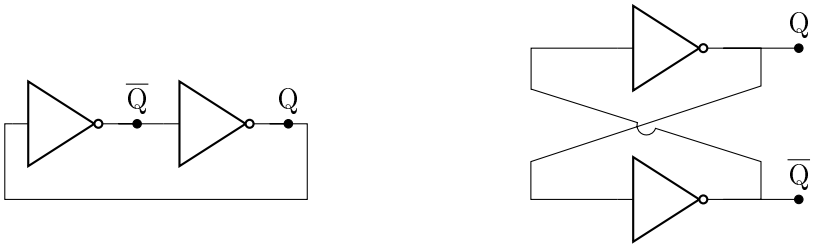
4 Logique séquentielle, mémorisation

Les bistables sont des circuits très employés en électronique numérique en raison de leur multiples applications. Leur première fonction est de mémoriser une information logique.

Les bistables sont des circuits **séquentiels**, c'est à dire que leur sortie dépend des entrées **et** de l'état précédent du système. Contrairement aux circuits combinatoires de la section précédente, l'état actuel du système dépend de son passé.

4.1 Bistable élémentaire

Ci-dessous est représenté sous 2 formes différentes le bistable le plus simple.

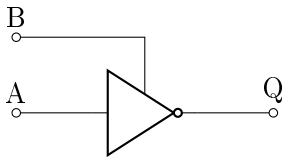


Ce bistable possède 2 états stables (d'où son nom) :
- soit Q est à l'état haut (1) et $\overline{Q}$ est à l'état bas (0)
- soit Q est à l'état bas (0) et $\overline{Q}$ est à l'état haut (1)

Ce circuit est donc une mémoire. Malheureusement, il est difficile de modifier son état autrement qu'en court-circuitant une des sortie à un état déterminé : il n'y a en effet pas d'entrée à ce circuit.

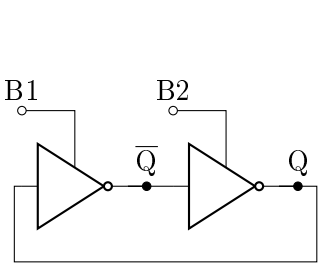
4.2 Amélioration

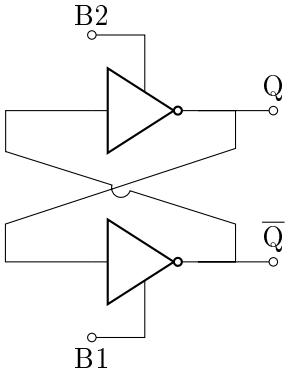
Imaginons que l'on dispose d'inverseurs dont on peut mettre la sortie dans un état déterminé. Par exemple l'inverseur suivant :



- si B est à l'état bas, alors Q est l'inverse de A (inversion normale de A)
- si B est à l'état haut, alors Q est à l'état bas (par exemple)

On peut alors réaliser le montage suivant :





On peut maintenant réaliser les opérations suivantes :

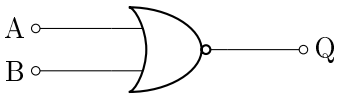
- mettre l'entrée B2 à l'état haut et l'entrée B1 à l'état bas : la sortie Q est alors mise à l'état bas.
- mettre l'entrée B2 à l'état bas et l'entrée B1 à l'état haut : la sortie Q est alors mise à l'état haut.
- mettre les 2 entrées B1 et B2 à l'état bas : le circuit reste dans sont état précédant : c'est la mémorisation.

Le cas où les 2 entrées B1 et B2 sont à l'état haut n'est pas intéressant : il force les 2 sorties Q et $\overline{Q}$ à l'état bas. C'est un cas indésirable.

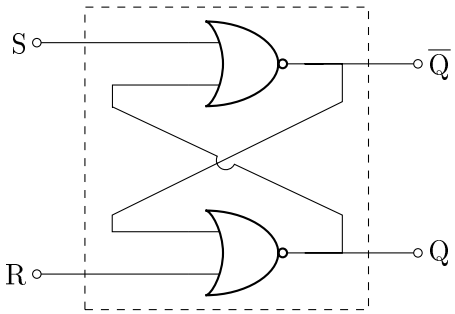
4.3 Bistable RS

En fait, on connaît, sans le savoir, des éléments logiques qui répondent à l'inverseur spécial vu ci-dessus.

Prenons par exemple la porte NOR (Non OU).



Cette porte répond parfaitement à la table de vérité de l'inverseur spécial : si son entrée B est à l'état haut, la sortie Q est forcée à l'état bas ; sinon, la sortie Q est l'inverse de l'entrée A. Construisons alors le bistable à partir de ces portes et renommons (S, R) les entrées (B1, B2).



— Pour mettre la sortie Q à l'état bas, il suffit d'appliquer un état haut à l'entrée R (Reset : mise à l'état bas) et un état bas à l'entrée S (Set : mise à l'état haut)
— Pour mettre la sortie Q à l'état haut, il suffit d'appliquer un état haut à l'entrée S (Set : mise à l'état haut) et un état bas à l'entrée R (Reset : mise à l'état bas)
— Pour être en mémorisation, il suffit d'appliquer un état bas aux 2 entrées.

Si on appelle l'état bas le niveau inactif et l'état haut le niveau actif, alors on peut dire :

— pour mettre le bistable dans un état déterminé, il suffit d'appliquer un niveau actif à l'entrée correspondante.
— pour mettre le bistable en mémorisation, il suffit de ne mettre aucun niveau actif
— l'état indésirable a lieu lorsqu'on demande au bistable en même temps une mise à l'état haut et une mise à l'état bas.

4.4 Table de vérité du bistable R-S

Si on n'est pas convaincu par les développements précédents, on peut écrire la table de vérité du bistable R-S :

R	S	Q_n	$\overline{Q_n}$	Q_{n+1}	$\overline{Q_{n+1}}$
0	0	0	0	?	?
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	impossible	
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	impossible	
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	impossible	
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	impossible	

En entrée de la table, on met les 2 sorties stables à l'instant t et les entrée que l'on applique à ce même instant. En sortie de la table, on détermine l'état des 2 sorties à l'instant $t + t_p$, c'est-à-dire après le temps de propagation du bistable. Cette table de vérité peut être simplifiée, en effet, les états impossibles ne sont pas à prendre en considération puisqu'ils ne seront jamais rencontrés. Il ne faut cependant pas confondre état impossible en entrée (lignes impossibles) et état indéterminé en sortie (ligne 1). La table devient :

R	S	Q_n	$\overline{Q_n}$	Q_{n+1}	$\overline{Q_{n+1}}$
0	0	0	0	?	?
0	0	0	1	0	1
0	0	1	0	1	0
0	1	X	X	1	0
1	0	X	X	0	1
1	1	X	X	0	0

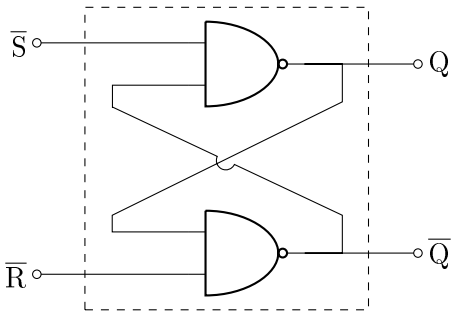
Le dernier état est évidemment indésirable puisqu'il rend égales des sorties normalement complémentaires (l'ordre donné au bistable est incohérent). Si on évite cet état, on peut supprimer la dernière ligne du tableau ainsi que la première. On peut alors simplifier cette table pour avoir :

R	S	Q_{n+1}	$\overline{Q_{n+1}}$
0	0	Q_n	$\overline{Q_n}$
0	1	1	0
1	0	0	1

De cette table, on retire heureusement les mêmes conclusions que précédemment. La méthode par table de vérité est assez lourde dans son entièreté. Il vaut donc mieux essayer d'établir directement la table simplifiée comme vu précédemment.

4.5 Réalisation du bistable R-S

Soit le circuit suivant :



Question 7. Quel est l'état des entrées qui assure la fonction de mémorisation ?

Question 8. Que faut il appliquer au circuit pour réaliser un Set ou Reset ?

Question 9. Quelle est l'anomalie des sorties lorsque les deux entrées sont à l'état bas ?

Question 10. Donner à chaque valeur des entrées ($\overline{R}$, $\overline{S}$) une appellation parmi les suivantes :

- mémorisation
- mise à 0
- mise à 1
- indésirable

Question 11. Justifier appellation R-S. (R : Reset S : Set) ; Quel est le niveau logique actif de ces 2 entrées ?

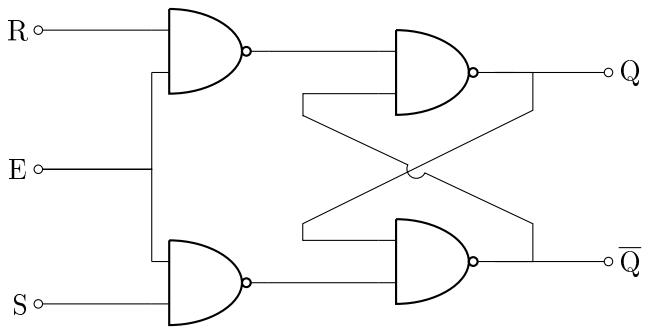
Question 12. Pourquoi parle-t-on d'entrées en logique inverse ?

Question 13. Que pourrait-il se passer si on passe de l'état [mise à 1] à l'état [mémorisation] en passant transitoirement par l'état [indésirable] ?

Question 14. Réaliser le circuit ci-dessus sur le protoboard et vérifier vos conclusions des questions 7 à 13.

4.6 Bistable R-S avec entrée d'activation (Enable)

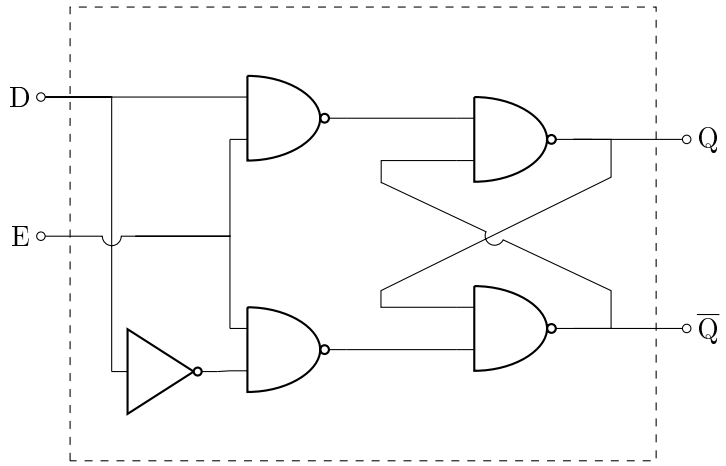
Modifier le schéma précédent comme suit :



Question 15. Mettre en évidence l'avantage de l'entrée enable du point de vue des modifications des entrées R et S.

4.7 D latch

Le D latch peut être réalisé avec le schéma suivant :



L'inverseur en tête du montage a pour effet de supprimer l'état illicite du bistable R-S de sortie.

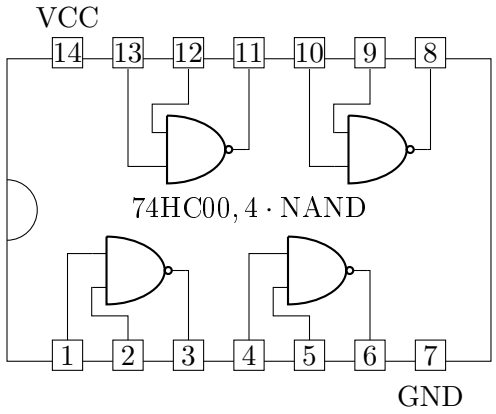
Question 16. Déterminer son fonctionnement (table de vérité simplifiée).

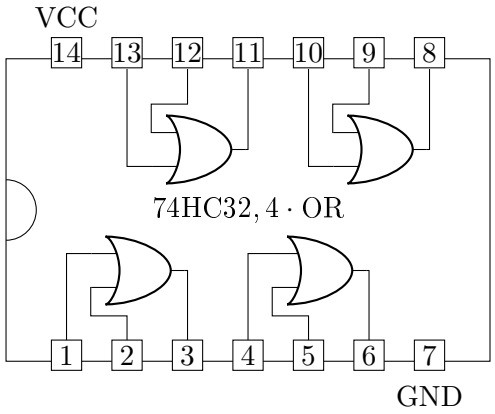
Question 17. Réaliser le D LATCH à l'aide de portes logiques. Le circuit 74HC04 contient 6 inverseurs.

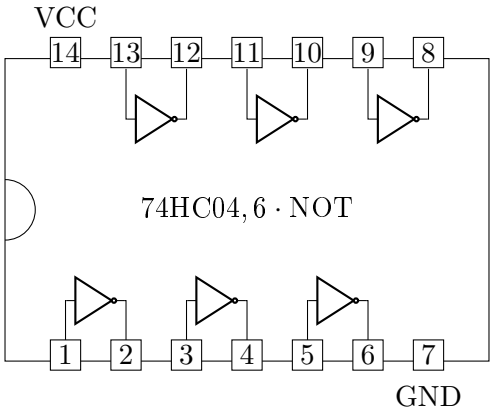
Question 18. Vérifier que votre circuit reproduit bien le fonctionnement attendu.

Question 19. Peut-on, à votre avis, modifier D n'importe quand ?

A Brochages







Pour information, les documentations complètes :

- 74HC00 (Quad NAND) https://www.fairchildsemi.com/datasheets/74/74VHC00.pdf
- 74HC04 (Hex NOT) https://www.fairchildsemi.com/datasheets/74/74VHC04.pdf
- 74HC32 (Quad OR) https://www.fairchildsemi.com/datasheets/74/74VHC32.pdf