

1. Présentation

L'Unité Arithmétique et Logique (ALU en anglais) intégré à une Unité Centrale de Traitement :

- réalise les opérations arithmétiques (addition, multiplication, ...), logiques (et, ou, ...), de comparaisons ou de déplacement de mémoire (copie de ou vers la mémoire),
- stocke les données dans des mémoires d'accès très rapide appelées **registres**.

Un circuit logique permet de réaliser une opération booléenne. Il prend en entrée un ou des signaux électriques, dans un état « haut » (symbolisé par un « 1 » ou « vrai ») ou un état « bas » (symbolisé par un « 0 » ou « faux ») et donne en sortie un ou des signaux électriques de même nature.

Il existe deux catégories de circuit logique :

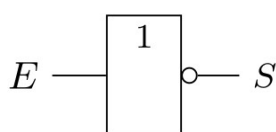
- les circuits **combinatoires** (les états en sortie dépendent uniquement des états en entrée)
- les circuits **séquentiels** (les états en sortie dépendent des états en entrée ainsi que du temps et des états antérieurs)

Ces opérations booléennes sont directement liées à l'**algèbre de Boole**.

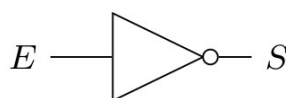
2. La porte NON

Le plus simple des circuits combinatoires est la porte « NON » qui inverse l'état en entrée : si l'entrée de la porte est dans un état « bas » alors la sortie sera dans un état « haut » et vice versa.

Si on symbolise l'état "haut" par un "1" et l'état "bas" pour un "0".



Symbole SI



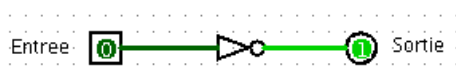
Symbole US

Table de vérité
Porte Non

E	S
0	1
1	0

Algèbre de Boole : $S = \bar{E}$ ou $S = \neg E$

2.1. **Lancer** le logiciel LogiSim. **Saisir** le circuit de la fonction NON.



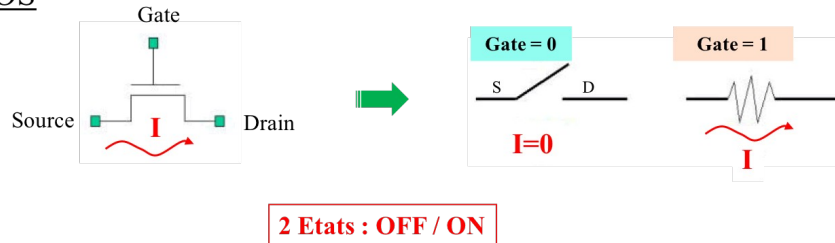
2.2. **Sélectionner** l'icône  dans les raccourcis pour activer la simulation.

2.3. **Cliquer** sur l'icône de l'entrée  et observer l'état de la sortie .

2.4. Valider la table de vérité de la porte NON

Les circuits logiques sont réalisés grâce à des transistors MOS. Ces derniers peuvent de type P ou de type N. La fonction NON est réalisable à l'aide de 2 transistors MOS (1 type N et 1 type P)

NMOS



PMOS



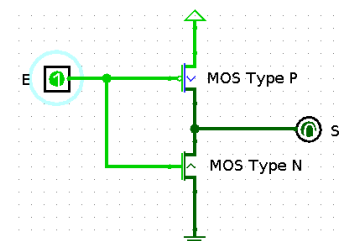
source : <https://openclassrooms.com/fr/courses/5439146-faites-vos-premiers-pas-dans-le-monde-de-l-electronique-numerique/5693601-implementez-une-porte-logique-a-base-de-transistors-mos>

2.5. Ouvrir le fichier « Porte_NON_MOS.circ » avec le logiciel LogiSim. Vérifier le fonctionnement attendu.

.....

.....

.....



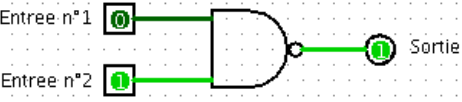
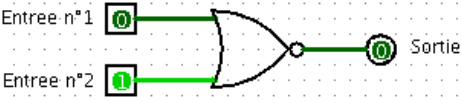
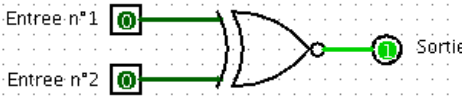
3. Portes élémentaires à deux entrées

Saisir les différents circuits combinatoires dans le logiciel LogiSim et relever leur table de vérité. Attention de préciser le nombre d'entrées de la fonction, ici 2 entrées E1 et E2 dans les paramètres de la porte.

Modifier le schéma pour réaliser la table de vérité des fonctions élémentaires à deux entrées (number of input) OU, OU-Exclusif, ET-NON, OU-NON et OU-Exclusif-Non. Proposer une conclusion.

Selection: XNOR Gate	
Facing	East
Data Bits	1
Gate Size	Medium
Number Of Inputs	2
Output Value	0/1
Label	

<p>Fonction ET (AND)</p> <p>Entree n°1</p> <p>Entree n°2</p> <p>Sortie</p> <p>Table de vérité</p> <table> <tr> <th>Entrée 1</th><th>Entrée 2</th><th>Sortie</th></tr> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </table> <p>$S = E1 \cdot E2$ ou $S = E1 \wedge E2$</p>	Entrée 1	Entrée 2	Sortie	0	0		0	1		1	0		1	1		<p>Fonction OU (OR)</p> <p>Entree n°1</p> <p>Entree n°2</p> <p>Sortie</p> <p>Table de vérité</p> <table> <tr> <th>Entrée 1</th><th>Entrée 2</th><th>Sortie</th></tr> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </table> <p>$S = E1 + E2$ ou $S = E1 \vee E2$</p>	Entrée 1	Entrée 2	Sortie	0	0		0	1		1	0		1	1		<p>Fonction OU Exclusif (XOR)</p> <p>Entree n°1</p> <p>Entree n°2</p> <p>Sortie</p> <p>Table de vérité</p> <table> <tr> <th>Entrée 1</th><th>Entrée 2</th><th>Sortie</th></tr> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </table> <p>$S = E1 \oplus E2$</p>	Entrée 1	Entrée 2	Sortie	0	0		0	1		1	0		1	1	
Entrée 1	Entrée 2	Sortie																																													
0	0																																														
0	1																																														
1	0																																														
1	1																																														
Entrée 1	Entrée 2	Sortie																																													
0	0																																														
0	1																																														
1	0																																														
1	1																																														
Entrée 1	Entrée 2	Sortie																																													
0	0																																														
0	1																																														
1	0																																														
1	1																																														

<p>Fonction ET-NON (NAND)</p>  <p>Table de vérité</p> <table border="1"> <thead> <tr> <th>Entrée 1</th><th>Entrée 2</th><th>Sortie</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </tbody> </table> <p>$S = \overline{E1} \cdot \overline{E2}$ ou $S = \neg (E1 \wedge E2)$</p>	Entrée 1	Entrée 2	Sortie	0	0		0	1		1	0		1	1		<p>Fonction OU-NON (NOR)</p>  <p>Table de vérité</p> <table border="1"> <thead> <tr> <th>Entrée 1</th><th>Entrée 2</th><th>Sortie</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </tbody> </table> <p>$S = \overline{E1} + \overline{E2}$ ou $S = \neg (E1 \vee E2)$</p>	Entrée 1	Entrée 2	Sortie	0	0		0	1		1	0		1	1		<p>Fonction OU Exclusif NON (XNOR)</p>  <p>Table de vérité</p> <table border="1"> <thead> <tr> <th>Entrée 1</th><th>Entrée 2</th><th>Sortie</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </tbody> </table> <p>$S = \overline{E1} \oplus \overline{E2}$ ou $S = \neg (E1 \oplus E2)$</p>	Entrée 1	Entrée 2	Sortie	0	0		0	1		1	0		1	1	
Entrée 1	Entrée 2	Sortie																																													
0	0																																														
0	1																																														
1	0																																														
1	1																																														
Entrée 1	Entrée 2	Sortie																																													
0	0																																														
0	1																																														
1	0																																														
1	1																																														
Entrée 1	Entrée 2	Sortie																																													
0	0																																														
0	1																																														
1	0																																														
1	1																																														


4. Algèbre de Boole

L'association des fonctions logiques élémentaires fait l'objet d'une algèbre particulière appelée « algèbre de Boole ». Vous disposez en annexe d'une ressource sur cette algèbre. Cette activité vous propose de les aborder à travers la simulation.

Procéder aux simulations pour les associations ci-dessous et **déterminer** la valeur de la sortie.


Rechercher le nom du théorème ou de la propriété pour chacun des cas proposés :

E . E = ...




.....

E . 0 = ...




.....

E . 1 = ...




.....

E + E = ...




.....

E + 1 = ...



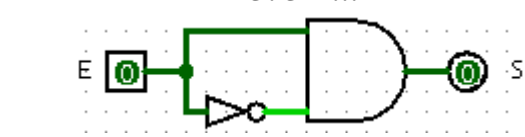
.....

E + 0 = ...



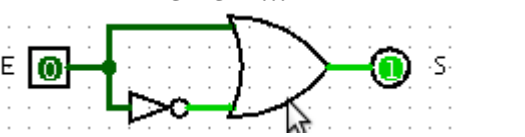
.....

e . e = ...



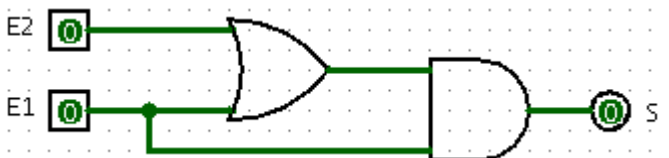
.....

e + e = ...




.....

E1 . (E1 + E2) = ...



.....

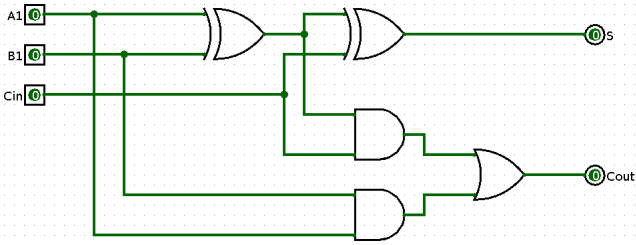
(E1 . E2) + E1 = ...



.....

5. Structure combinatoire complexe

Le jeu d'instructions d'une structure ALU d'un CPU comporte des opérations simples (AND, OR, ...) réalisables avec des portes élémentaires, mais également d'opérations plus complexes (ADD, SUB, MUL,...) obtenues grâce à des associations de fonctions élémentaires en structures plus complexes.



Ouvrir le fichier « add_2x1bits.circ » dans le logiciel LogiSim. Compléter la table de vérité ci-dessous et préciser selon vous le rôle de ce circuit combinatoire.

E1	E2	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1			
0	1			
1	0			
1	0			
1	1			
1	1			

.....
.....
.....

Ouvrir le fichier « add_2x2bits.circ » dans LogiSim. Préciser selon vous le rôle de ce circuit combinatoire.

.....
.....

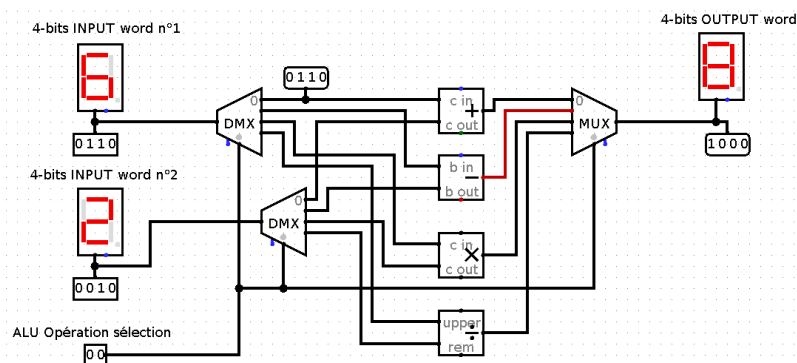
A l'aide du lien suivant <https://fr.wikipedia.org/wiki/Additionneur>, préciser quel type d'additionneur est proposé dans le circuit logique combinatoire précédent. Critiquer la solution : Est-ce ainsi qu'est physiquement réalisée les additionneurs au sein des ALUs et pourquoi ?

.....
.....
.....
.....
.....
.....

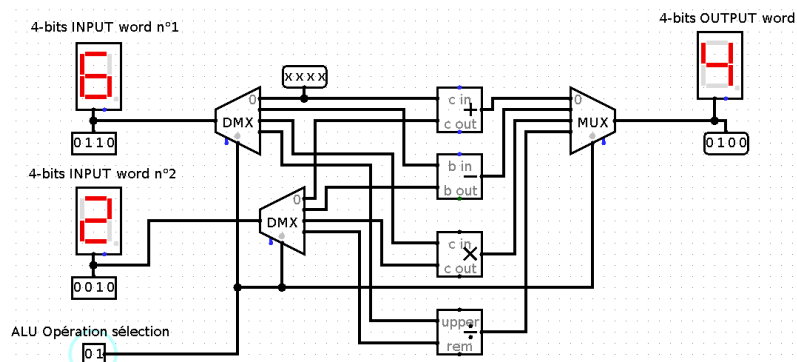
L'ALU est un composant complexe qui permet de choisir l'opération à réaliser sur 1 ou plusieurs registres.

Le fichier « ALU_Basic.circ » est une simulation basique d'une ALU, possédant 4 opérations élémentaires (addition, soustraction, multiplication, division), sélectionnables à partir du mot de 2 bits « ALU sélection operation », agissant sur 2 mots de données de 4 bits « word n°1 » et « word n°2 ». Cet ALU ne gère pas les dépassement ou erreur ...

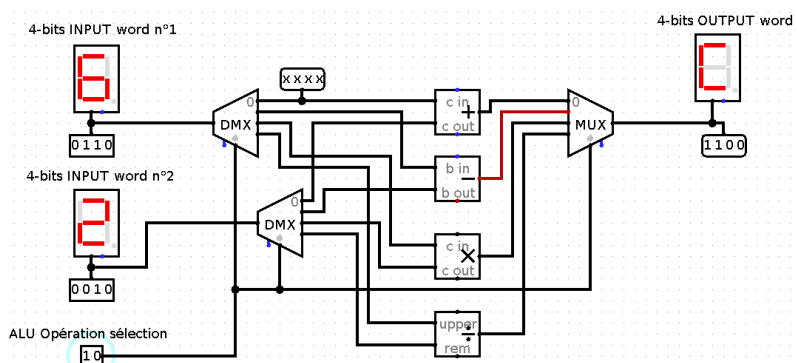
Ouvrir ce fichier et **valider** le fonctionnement attendu :



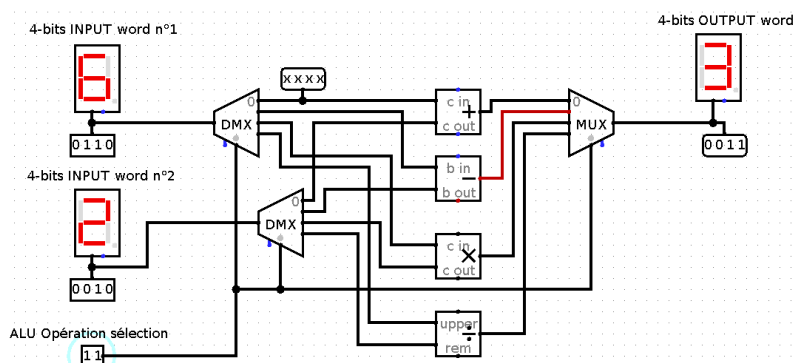
ALU / Addition



ALU / Soustraction



ALU / Multiplication



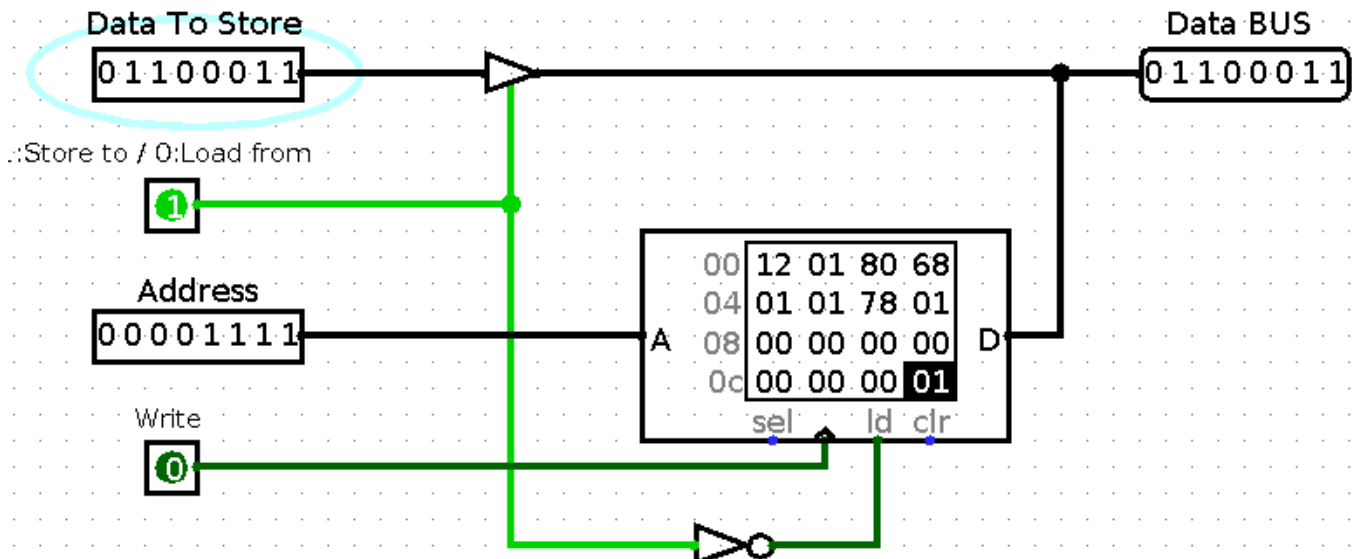
ALU / Division

6. Mémoire RAM

La mémoire peut se concevoir comme une série de cellules, chaque cellule étant capable de stocker 1 octet. Chacune de ces cellules possède une adresse. Les opérations sur la mémoire sont de 2 types : lecture / écriture. Une opération de lecture consiste à aller lire l'octet situé à une adresse mémoire et une opération d'écriture consiste à écrire un octet donné à une adresse mémoire.

Il n'est pas nécessaire de passer par une cellule pour accéder à une autre (ce qui formerait une mémoire à accès séquentiel). Cette faculté d'accéder directement à chaque cellule explique le nom RAM : Random access memory, mémoire à accès arbitraire.

Les mémoires sont obtenues à l'aide de circuits séquentiels, hors d'étude ici. Nous nous limiterons au module RAM pré-disponible du logiciel LogiSim.



Pour lire une donnée depuis la mémoire :

- positionner le bit « Store to / Load from » à zéro,
- renseigner l'adresse mémoire de la donnée à récupérer
- la donnée mémoire est alors disponible sur le bus de données (coté droite)
- dans ce mode, chaque changement d'adresse provoque la copie de la case mémoire vers « Data Bus »

Pour écrire une donnée dans la mémoire :

- positionner le bit « Store to / Load from » à un,
- renseigner l'adresse mémoire de la donnée à récupérer
- saisir la donnée à écrire sur le bus dans le champ « Data to store »,
- envoyer un front montant (passage de 0 à 1) sur le bit « write »,
- dans ce mode, la valeur du « Data Bus » est constante et vaut la donnée à écrire.

À partir des instructions ci-dessus, **écrire** en mémoire les valeurs 0x18, 0xFF, 0x11 respectivement aux adresses 0x03, 0x05 et 0x09

Le logiciel n'autorise pas ce fonctionnement, mais le mode opératoire peut s'envisager : Comment recopier la donnée contenue à l'adresse 0x5 à l'adresse mémoire 0x04 ?

Annexe : Algèbre de Boole

L'algèbre booléenne se base sur **les deux règles suivantes** :

- **Les opérations** sont aux nombres de 3 classées de la moins prioritaire à la plus prioritaire :
 - x L'addition binaire équivalente à une fonction logique OU noté « + »
 - x Le produit binaire équivalent à une fonction logique ET noté « . »
 - x La complémentation notée « $\bar{}$ »
- **Les variables** possèdent deux états notés **0** ou **1** définis ainsi : $\bar{0} = 1$ et $\bar{1} = 0$

Propriétés des opérations booléennes

- **Opérateur involutif** : $\bar{\bar{a}} = a$ ou $\neg(\neg a) = a$
- **La commutativité** : $a + b = b + a$
- **L'associativité** : $(a + b) + c = a + (b + c)$
- **Élément neutre** : $a + 0 = a$ et $a \cdot 1 = a$
- **Élément absorbant** : $a \cdot 0 = 0$ et $a + 1 = 1$
- **La distributivité** : $a + (b \cdot c) = (a + b) \cdot (a + c)$ et $a \cdot (b + c) = a \cdot b + a \cdot c$

Attention, contrairement à l'algèbre classique, **la somme booléenne est distributive !!!**

Les théorèmes applicables

- **Somme d'une variable et de son complément** : $a + \bar{a} = 1$
- **Produit d'une variable et de son complément** : $a \cdot \bar{a} = 0$
- **Idempotence** : Produit d'une variable avec elle même : $a \cdot a = a$
- **Idempotence** : Somme d'une variable avec elle même : $a + a = a$
- **Absorption** : $a + a \cdot b = a$ et $a \cdot (a + b) = a$
 $\bar{a} + a \cdot b = \bar{a} + b$ ou encore $a + \bar{a} \cdot \bar{b} = a + \bar{b}$
- **Le théorème de Morgan** :
 - x Le complément d'une somme de variables est égale aux produits des compléments de ces variables : $\overline{a + b + c + d} = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d}$
 - x Le complément d'un produit de variables est égale à la somme des compléments de ces variables : $\overline{a \cdot b \cdot c \cdot d} = \bar{a} + \bar{b} + \bar{c} + \bar{d}$