Math93.com

Circuits logiques

1 Définitions

Définition 1

Toutes les **expressions booléennes** peuvent être représentées par des **circuits logiques**, aussi appelés circuits combinatoires. Les opérations élémentaires sur les booléens sont représentées par des portes logiques.

1.1 Porte NON notée : NON, not, $\backsim$, $\overline{V} = F$, $\neg$

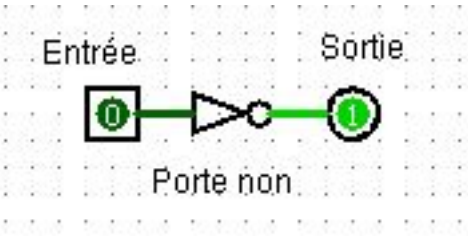


Table de vérité : loi NOT

Entrée	Sortie
a	$\overline{a} = non(a) = \neg(a)$
0	1
1	0

Autres notations : $not(a)$, $\backsim a$

1.2 La porte ET notée : ET, and, & , $\wedge$, conjonction logique $(\times)$

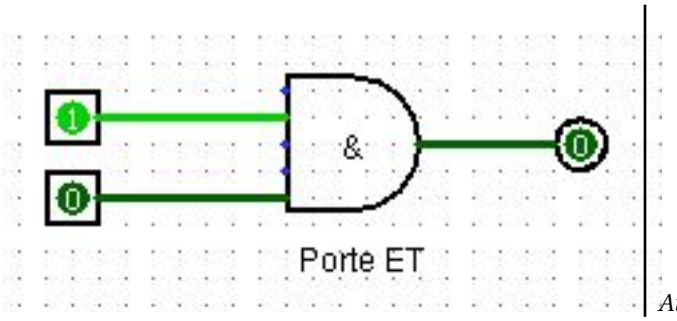


Table de vérité : loi AND

Entrées		Sortie
a	b	a and $b = a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

Autres notations : $a\&b$

1.3 La porte OU notée : OU, or, | , $\vee$, disjonction $(+)$

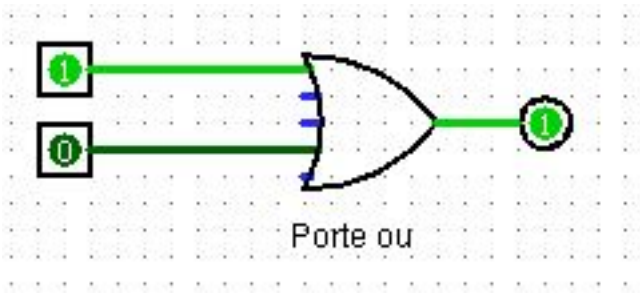


Table de vérité : loi OU

Entrées		Sortie
a	b	a or $b = a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

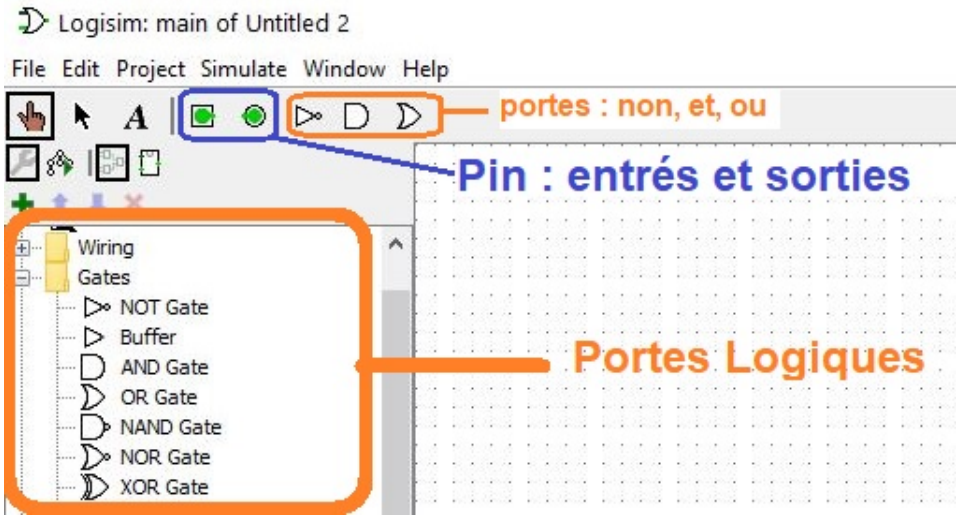
Autres notations : $a \mid b$, a *or* b , a *OR* b

2 Les circuits logiques sur le logiciel Logisim : un exemple

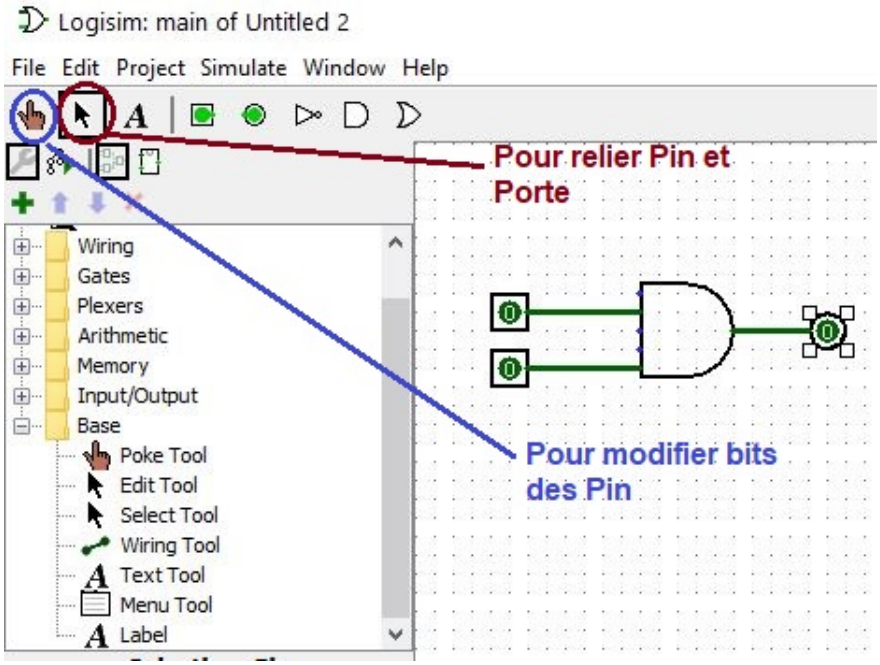
Téléchargement sur : http://www.cburch.com/logisim/

— **Étape 1**
Ouvrir le logiciel Logisim. C'est un simulateur booléen. Il ne cherche pas à faire intervenir des valeurs autres que 0 ou 1. *Il s'agit d'un logiciel libre et gratuit.*

— **Étape 2**
On découvre une feuille blanche sur laquelle on peut réaliser des circuits logiques ainsi qu'un menu qui propose plusieurs composants.
Dans le dossier *Gates*, trouvez une porte logique AND. Sélectionnez-la et dessinez-la sur la feuille blanche par simple clic.



— **Étape 3**
On utilise les "*Pin*" carrés verts pour les entrées et le "*Pin*" disque vert pour la sortie.
Relier les entrées et la sortie à la porte en cliquant sur la flèche.



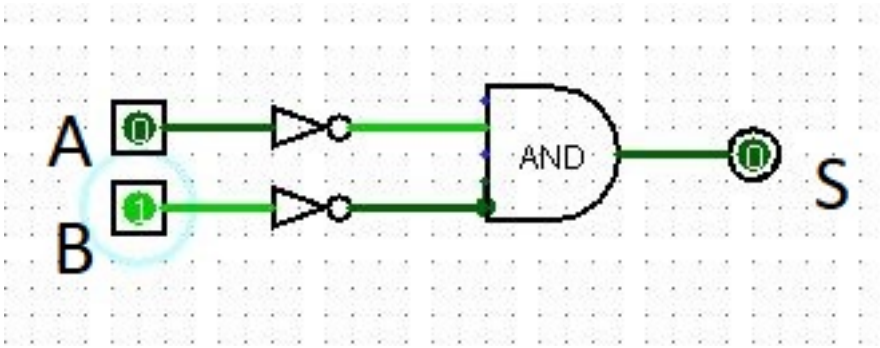
— **Étape 4**
On peut modifier les valeurs d'entrée en cliquant sur les "*Pin*" avec la "*Main*" du menu.
Sélectionner le mode Simulation (la main en haut à gauche). En cliquant sur les entrées, relever la valeur de sortie et vérifier que la table de vérité est correcte.
Aide : *Menu Projet / Analyse circuit / Table ...*

3 Les circuits logiques sur le logiciel Logisim : exercices

3.1 Pour bien débuter

Exercice 3.1 - Porte NOR

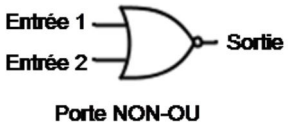
1. Réaliser le circuit suivant à l'aide du logiciel Logisim.



2. Donner l'expression booléenne de S en fonction des variables A et B.
3. Compléter la table de vérité ci-dessous.

A	B	S
0	0	
0	1	
1	0	
1	1	

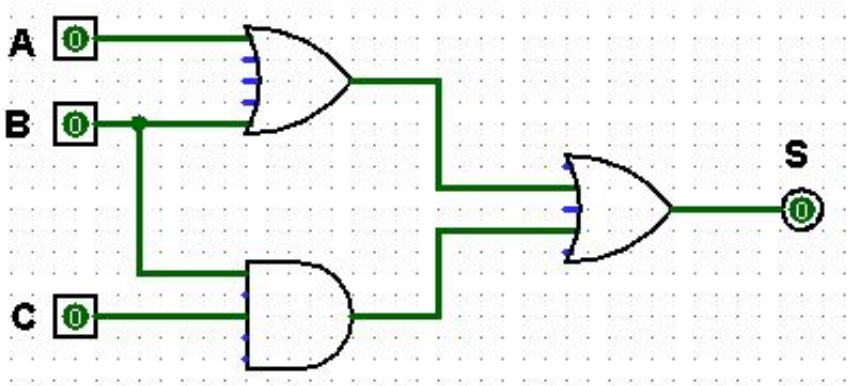
4. Par quel circuit comprenant seulement deux portes peut-on remplacer le circuit étudié?
5. Ce circuit est en fait celui de la porte NOR. Vérifier cela en utilisant la porte NOR de logisim.



Porte NON-OU

Exercice 3.2

On considère le circuit logique ci-dessous.



1. Donner l'expression booléenne de S en fonction des variables A, B et C.
2. Compléter la table de vérité ci-dessous.

A	B	C	S
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

3. En déduire une formule pour S qui ne dépend que des variables A et B.

3.2 Porte XOR

Exercice 3.3 - Porte XOR : disjonction (OU) exclusive $\oplus$

La fonction logique OU eXclusif (XOR) est définie de la manière suivante :

A xor B est Vrai

si et seulement si

A Vrai or B Vrai mais pas les deux.

La fonction booléenne de passage s'écrit :

$$f(A,B) = A \oplus B = A \texttt{ xor } B$$

Porte OU EXCLUSIF

1. Écrire la table de vérité de la loi XOR.

Table de vérité : loi XOR

Entrées		Sortie
a	b	$a \oplus b = a \texttt{ xor } b$
0	0	
0	1	
1	0	
1	1	

2. Écrire la table de vérité de la loi définie par :

$$f(A,B) = (\texttt{not } A \texttt{ and } B) \texttt{ or } (A \texttt{ and not } B) = \left(\overline{A} \texttt{ and } B\right) \texttt{ or } \left(A \texttt{ and } \overline{B}\right)$$

Table de vérité

Entrées		Sortie
A	B	$\left(\overline{A} \texttt{ and } B\right) \texttt{ or } \left(A \texttt{ and } \overline{B}\right)$
0	0	
0	1	
1	0	
1	1	

3. Concevoir un circuit qui, étant données deux entrées A et B, donne en sortie la valeur A xor B, en utilisant seulement des portes Non, Et, Ou. Puis directement avec la porte XOR.

Aide

La question précédente a montré que :

$$A \texttt{ xor } B = (\texttt{not } A \texttt{ and } B) \texttt{ or } (A \texttt{ and not } B) = A \oplus B = \left(\overline{A} \texttt{ and } B\right) \texttt{ or } \left(A \texttt{ and } \overline{B}\right)$$

Remarque

Application en électricité domestique.
Une application utilisée de l'opérateur logique XOR en électricité domestique est dans les salles où une ampoule peut être allumée ou éteinte par deux interrupteurs placés près de deux entrées. Chacun des deux interrupteurs peut soit allumer ou éteindre l'ampoule quelle que soit la position de l'autre interrupteur. Pour obtenir une telle fonctionnalité, on doit brancher les deux interrupteurs afin de former un opérateur XOR. C'est le montage dit « va-et-vient ».

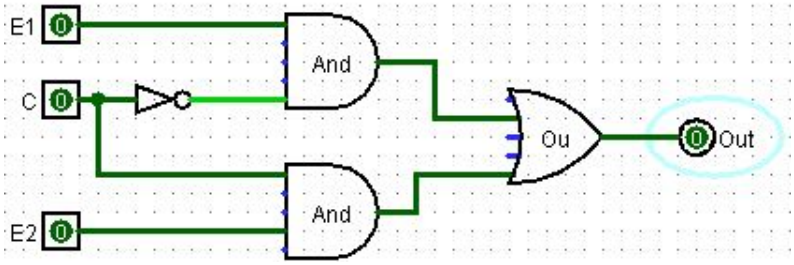
3.3 Circuit Multiplexeur

Définition 2

Un **multiplexeur** (abréviation : MUX) est un circuit permettant de concentrer sur une même voie de transmission différents types de liaisons (informatique, télécopie, téléphonie, télétex) en sélectionnant une entrée parmi N. Il possédera donc une sortie et N entrées, ainsi qu'une entrée de commande de plusieurs bits permettant de choisir quelle entrée sera sélectionnée.
Il sert d'accès aux réseaux de transmission de données numériques ou analogiques.

Exercice 3.4 - Circuit MUX-2

On considère le circuit logique suivant.



1. Donner l'expression de Out en fonction de E_1, E_2 et C.
2. Compléter le tableau de vérité de ce circuit.

C	E_1	E_2	Out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Remarque

Le circuit étudié est appelé **multiplexeur à 2 entrées**.
Selon la valeur de la commande (C), il permet de reproduire en sortie (Out) :

— le signal E_1 si C est à 0.

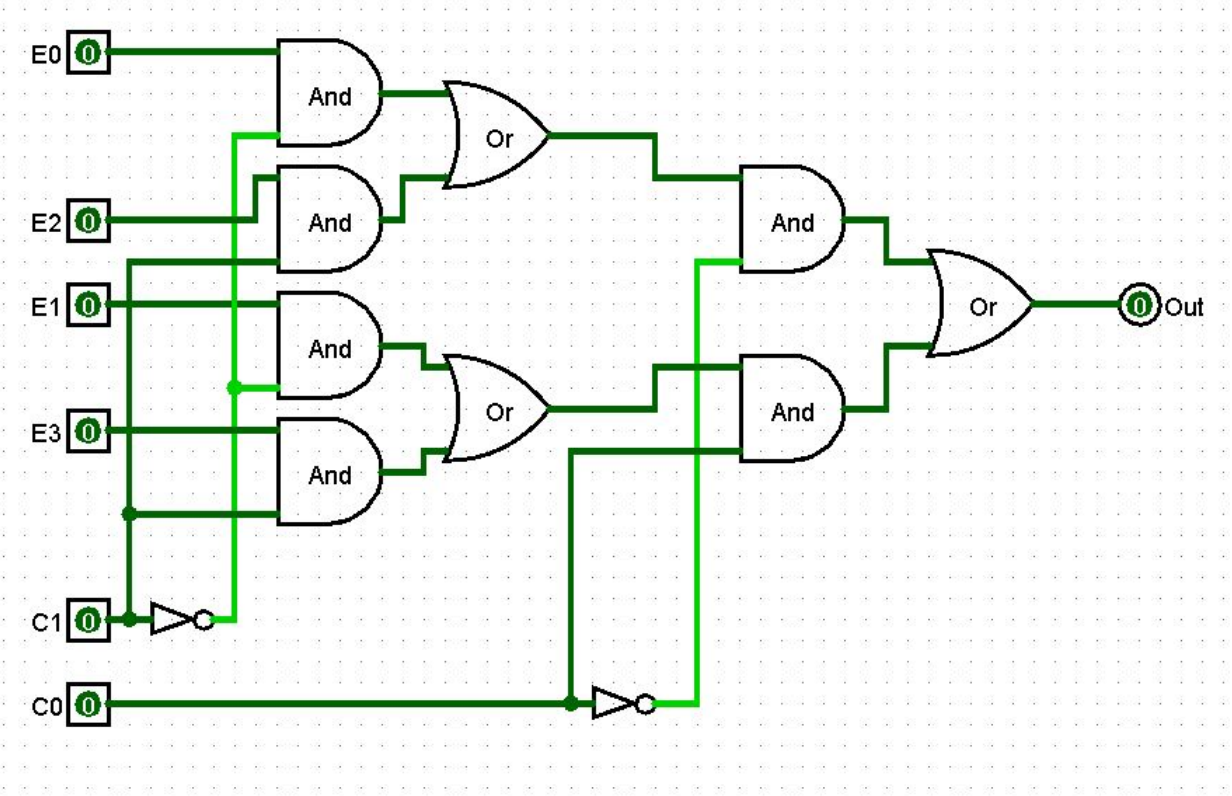
— le signal E_2 si C est à 1.

Exercice 3.5 - Circuit MUX-4 (difficile *)

Remarque

La premiére question est un peu longue. Vous pouvez passer l'exercice et y revenir s'il vous reste du temps.

On considère un multiplexeur à 4 entrées, dont le circuit est représenté ci-dessous.



Aide

Pour cet exercice, vous pourrez vous aider du logiciel logisim qui vous fournit directement la table de vérité du circuit.
Menu Project / Analyse Circuit / Table (et imput pour régler l'ordre des variables).

1. Par analyse du circuit, déterminer l'expression booléenne de Out en fonction des entrées E_0, E_1, E_2, E_3 et des commandes C_0 et C_1.
   On peut par exemple sur logisim vérifier quelle est la l'entrée reproduite en sortie selon les valeurs de C_0 et C_1.
2. Quelles sont les valeurs des commandes C_0 et C_1 qui permettent de sélectionner en sortie (Out) :
   - l'entrée E_0 ?
   - l'entrée E_1 ?
   - l'entrée E_2 ?
   - l'entrée E_3 ?

Remarque

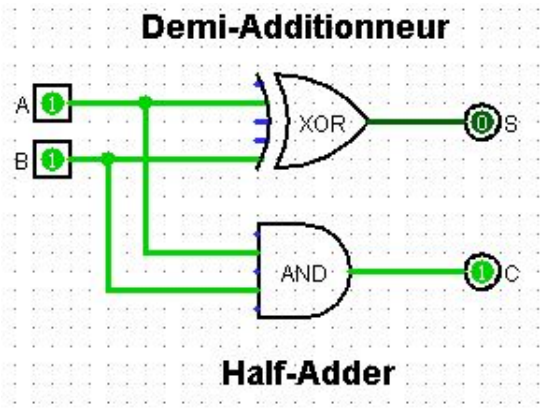
Le circuit étudié est appelé **multiplexeur à 4 entrées**.
Selon la valeur des commandes C_0 et C_1, il permet de reproduire en sortie (Out) le signal E_0, E_1, E_2 ou E_3. Voir la dernière question de l'exercice.

3.4 Les additionneurs ("*adders*")

3.4.1 Le demi-additionneur (*half adder*)

Le circuit étudié, appelé **demi-additionneur**, permet d'additionner deux bits A et B.
Il comporte deux sorties C et S qui représentent deux expressions booléennes.

Exercice 3.6



1. Donner les expressions booléennes de C et S en fonction de A et B.
2. Compléter la table de vérité de C et S.

Table de vérité : S

Entrées		Sortie
a	b	S
0	0	
0	1	
1	0	
1	1	

Table de vérité de C

Entrées		Sortie
a	b	C
0	0	
0	1	
1	0	
1	1	

3. Quel est le rôle des sorties C et S dans la fonction du circuit?

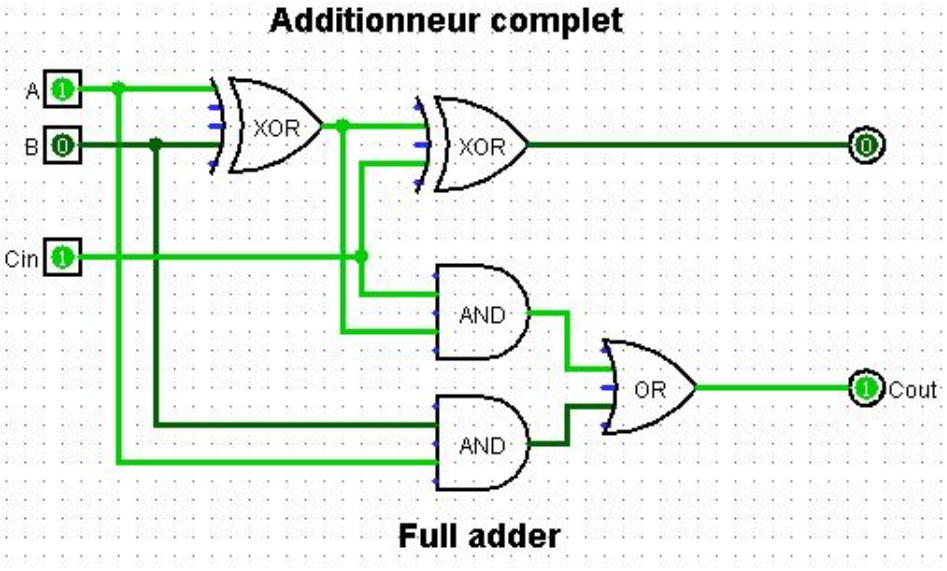
Remarque

Le choix de la lettre C vient du fait qu'en anglais, " **retenue** " se dit "**carry**" .

3.4.2 L'additionneur complet (*full adder*)

En pratique, une addition binaire est une suite d'additions sur 1 bit. Néanmoins, il faut connaître la retenue pour enchaîner ces additions. On réalise alors le circuit ci-dessous, appelé additionneur complet.
Il comporte deux sorties C_{out}, S et trois entrées, le bit A, le bit B et la retenue précédente C_{in}

Exercice 3.7



1. Réaliser ce circuit à l'aide du logiciel Logisim.
2. Compléter la table de vérité ci-dessous.

C_{in}	A	B	C_{out}	S
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

3.5 De la fonction au circuit

Exercice 3.8

On donne ci-dessous la table de vérité d'une expression booléenne $f(A,B)$.

A	B	$S = f(A,B)$
0	0	1
0	1	1
1	0	1
1	1	0

Retrouver l'expression de $f(A,B)$ en fonction de A et B puis construire le circuit associé .

Exercice 3.9

On reprend l'exercice 3.8 du TD sur les variables booléennes.
On donne ci-dessous les tables de vérité de différentes expressions booléennes U, V et W.
Retrouver les expressions de U, V et W en fonction de A et B et C puis construire les circuits associés.

A	B	C	U
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A	B	C	V
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

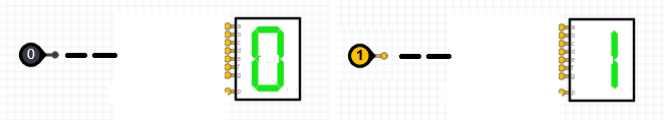
A	B	C	W
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Exercice 3.10

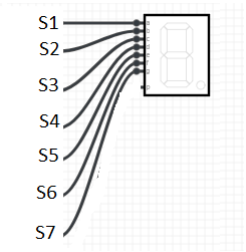
Affichage de 0 ou 1 outils :

- Afficheur à 7-segments display
- pin (entrée 0 ou 1)
- Des portes logiques

Objectif : A partir d'une entrée x à 1 bit faire afficher un 0 si on rentre 0 et affiche 1 si on rentre 1 comme sur cette exemple :



1. Placer un 7-segments display et un pin et tester le 7-segment pour connaître le rôle de chaque entrée du 7-segment.



2. Remplir la table de vérité suivante et Écrire les fonctions booléennes pour chaque entrée du 7-segment pour répondre à l'objectif.

x	S1	S2	S3	S4	S5	S6	S7
0							
1							

3. En déduire les fonctions booléennes en fonction de x :
   $S1 =$ $S2 =$ $S3 =$ $S4 =$ $S5 =$ $S6 =$ $S7 =$
4. Créer le circuit pour répondre à l'objectif

↓