

TP : Logic Friday

I. Pas de chameau aujourd'hui

1.a) Présentation de la séance

Aujourd'hui, nous allons mettre en application les différentes notions vues la semaine dernière sur les booléens et les portes logiques.

Cette séance sera un peu particulière : contrairement aux précédents TPs, nous n'utiliserons pas OCaml, mais du papier et des crayons, ainsi qu'un petit logiciel nommé *Logic Friday* qui permet de tracer et de tester des circuits booléens.

Pour commencer, lancez *Logic Friday* et référez-vous à la section suivante pour prendre en main ce nouvel outil.

1.b) Utiliser *Logic Friday*

Logic Friday est un logiciel assez complet pour l'étude des circuits booléens. Nous nous concentrerons dans ce TP sur l'outil de tracé de portes logiques, mais n'hésitez pas à explorer ses autres fonctionnalités.

Installation du logiciel *Logic Friday*

Attention, vous aurez sans doute besoin de droits d'administrateur pour procéder à cette installation.

Le logiciel *Logic Friday* est disponible à l'adresse suivante : <http://sontrak.com/>

Pour récupérer l'installateur, ouvrez la section Downloads et cliquez sur le lien Download *Logic Friday*. Lancez ensuite le fichier que vous venez de télécharger, et laissez-vous guider.

Une fois l'installation terminée, un raccourci a normalement été ajouté sur votre bureau. Double-cliquez sur ce lien pour lancer *Logic Friday*.

Créer un nouveau "diagramme de portes"

Pour commencer, vous allez créer ce qu'on appelle un "diagramme de portes", c'est-à-dire une zone de dessin dans laquelle on va pouvoir ensuite tracer des portes logiques.

Pour cela, cliquez sur [File / New / Gate Diagram](#) ou sur l'icône symbolisant une page blanche et sélectionnez [Gate Diagram](#).

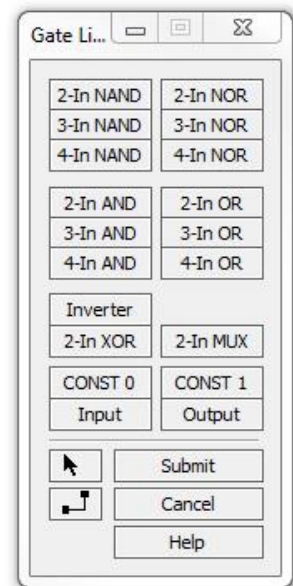
La fenêtre des composants

Une petite fenêtre s'ouvre, avec la liste des composants qu'on va pouvoir tracer sur ce nouvel espace de travail.

On y retrouve tous les éléments vus en cours :

- Constante Vrai : [CONST 0](#)
- Constante Faux : [CONST 1](#)
- Variable d'entrée : [Input](#)
- Variable de sortie : [Output](#)
- Porte NON : [Inverter](#)
- Porte ET : [2-In AND](#)
- Porte OU : [2-In OR](#)
- Etc.


La plupart des portes sont sur le format [X-In Y](#), où X est le nombre d' "entrées" de la porte (ce nombre valait toujours 2 dans les exemples vus en cours, mais ce n'est pas toujours le cas), et Y est le nom anglais de l'opération effectuée.



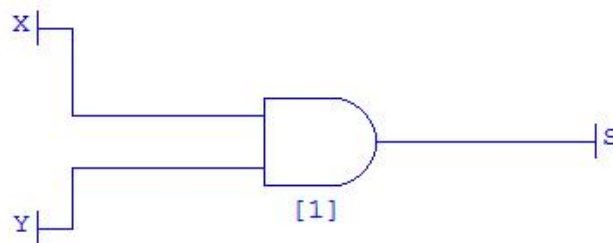
Ajouter des éléments

Pour ajouter un composant sur votre espace de travail, cliquez simplement sur la case correspondante de la fenêtre des composants, déplacez votre curseur sur l'espace de travail, et cliquez là où vous voulez déposer l'élément.

Dans le cas d'une variable d'entrée ou de sortie, vous pourrez en outre spécifier le nombre de cette variable (une fenêtre s'ouvrira automatiquement à cet effet).

Pour relier deux composants entre deux, vous pouvez ajouter un fil avec l'outil : cliquez sur  et cliquez ensuite sur les extrémités des composants que vous souhaitez relier.

Réalisez pour commencer le circuit suivant :



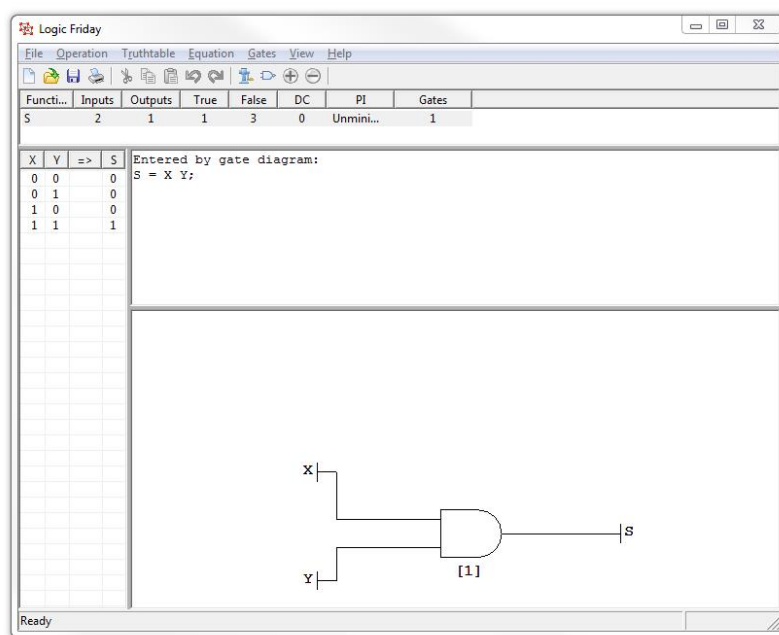
Calculer et afficher la table de vérité

Un des gros atouts de *Logic Friday* est qu'il permet de calculer automatiquement la table de vérité associée à un circuit booléen.

Pour cela, cliquez simplement sur [Submit](#) une fois votre circuit terminé. Si votre circuit est correct, la fenêtre principale sera alors divisée en plusieurs zones, et vous verrez apparaître à gauche la table de vérité.

Par défaut, seuls les cas de figure où la sortie vaut 1 sont affichés. Pour afficher la table de vérité dans son intégralité, cliquez sur [Truthtable / Show all rows](#).

Avec le circuit précédent, vous devriez obtenir :

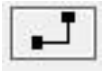


Modifier le diagramme de portes

Pour modifier le diagramme de portes, cliquez sur [Gates / Modify Gate Diagram](#).

Si vous souhaitez déplacer un élément, utilisez l'outil flèche : 

Pour supprimer un élément, sélectionnez-le avec la flèche et appuyez sur la touche Suppr de votre clavier. Attention, les fils qui relient les composants entre eux sont en plusieurs parties : pensez à bien tout supprimer si vous modifier ces liaisons.

En outre, vous aurez sans doute besoin de faire partir plusieurs fils du même point : pour cela,  cliquez sur l'outil Fil, puis cliquez sur l'extrémité dont vous voulez partir (même si un autre fil en part déjà), et cliquez sur l'autre extrémité de la liaison voulue.

Remarques

Attention, il est facile de perdre un diagramme de portes non enregistré. Pour enregistrer un diagramme, il vous suffit de faire [Submit](#) et de cliquer sur l'icône symbolisant une disquette¹.

Si vous appuyez sur la touche Echap pendant que vous éditez votre diagramme de portes, vous annulerez tous les changements depuis la dernière soumission (bouton [Submit](#)). Cela peut être pratique si vous avez cassé des choses sur votre circuit, mais vous pouvez aussi tout perdre si vous appuyez sur Echap sans avoir jamais soumis votre travail.

Enfin, tant que vous êtes en mode édition, tous les menus ou presque sont grisés : pour pouvoir accéder à ces menus, il vous faut soumettre votre travail. Pensez-donc à enregistrer votre travail à chaque fois que vous soumettez une nouvelle version.

II. Premiers exercices

2.a) Porte XOR

Pour commencer, tracez le circuit correspondant à une porte XOR pour deux variables d'entrées ("soit l'une, soit l'autre, mais pas les deux"), sans bien sur utiliser la porte XOR déjà existante.

Conseil : réfléchissez d'abord avec une feuille de papier et un crayon, et utilisez ensuite *Logic Friday* pour vérifier ensuite votre idée.

2.b) Porte ET sans porte ET

Cherchez ensuite comment réaliser une porte ET pour deux variables d'entrées sans utiliser les portes AND déjà présentes.

Indice : C'est juste un exercice pour vous faire réfléchir sur le fonctionnement des différentes portes, donc ce n'est pas grave si le résultat est inutilement compliqué.

2.c) Porte XOR juste avec des NAND *

Pour finir, imaginez un circuit pour réaliser une porte XOR à deux entrées en n'utilisant que des portes NAND.

Indice : La fonction XOR est symétrique.

¹ Sauvegardez vos fichiers dans un répertoire où vous avez les droits suffisants, comme par exemple "Mes Documents".

III. Ecriture en binaire

3.a) Rappels

On a vu en cours que tout nombre peut être décomposé de manière unique de la façon suivante :

$$x = \sum_{i=0}^{\infty} x_i \cdot 2^i \quad \text{avec} \quad 0 \leq x_i \leq 1$$

Tout nombre peut donc s'écrire avec uniquement des 0 et des 1 de la manière suivante :

$$x = \underline{x_n \cdots x_1 x_0}_2$$

Par exemple, $8 = \underline{1000}_2$, $13 = \underline{1101}_2$ et $2 = \underline{10}_2$

3.b) De la base 10 à la base 2

Pour vous familiariser avec ces notions, vous allez imaginer un circuit qui réalise cette conversion en calculant la valeur des x_i

Plus précisément, tracez le circuit C_1 suivant :

- C_1 a 8 variables d'entrée, numérotées de 0 à 7
- C_1 a 3 variables de sortie, nommées x_0 , x_1 et x_2
- On n'activera qu'une seule entrée de C_1 à la fois
- Si l'entrée x est activée, on doit avoir $x = \underline{x_2 x_1 x_0}_2$

Indice : Vous pouvez commencer avec 4 variables d'entrées et 2 variables de sorties.

3.c) De la base 2 à la base 10

Réciproquement, dessinez ensuite le circuit inverse, qui passe de l'écriture en binaire au nombre correspondant en base 10.

Plus précisément, dessinez le circuit C_2 suivant :

- C_2 a 3 variables d'entrée, nommées x_0 , x_1 et x_2
- C_2 a 8 variables de sortie, numérotées de 0 à 7
- Une seule sortie doit pouvoir être vraie à la fois
- L'unique sortie x activée est telle que $x = \underline{x_2 x_1 x_0}_2$

Indice : Vous pouvez commencer avec 2 variables d'entrées et 4 variables de sorties.

IV. Additionneur

4.a) Demi additionneur

Pour commencer, dessinez un circuit réalisant l'addition de deux booléens (c'est-à-dire de deux éléments pouvant valoir 0 ou 1), aussi appelé demi-additionneur.

Indice : Demandez-vous quelles sont les valeurs possibles en sortie, et de combien de variables de sortie vous aurez donc besoin.

4.b) Additionneur complet

Comme vous avez pu le constater avec le demi-additionneur, il est possible qu'une retenue apparaisse lors d'une addition.

Ainsi, la prochaine addition qu'on effectuera prendra en entrée non pas 2 booléens mais 3 : les deux variables d'entrées à additionner, et l'éventuelle retenue de l'opération précédente.

Construisez le circuit correspondant, appelé additionneur complet, qui fait la somme de deux booléens x et y ainsi que d'une retenue.

Indice : Encore une fois, demandez-vous quelles sont les valeurs possibles en sortie, et de combien de variables de sortie vous aurez donc besoin.

4.c) Additionneurs en parallèle

Pour finir, combinez deux additionneurs complets pour réaliser l'addition entre deux nombres écrits chacun sur 2 bits.

Une fois que vous aurez saisi le principe, vous pourrez en déduire aisément comment additionner deux nombres écrits chacun sur 3 bits, 32 bits ou 64 bits.

Indice : Essayez de faire à la main les additions entre deux nombres écrits en binaire :

$$\begin{array}{r} 6 \\ + 3 \\ \hline = ? \end{array} \quad \leftrightarrow \quad \begin{array}{r} 0110_2 \\ + 0011_2 \\ \hline = ? \end{array}$$

V. Comparateurs

5.a) Comparateur 1-bit

Dessinez un circuit avec deux variables d'entrées A et B et renvoyant vrai si et seulement si A est plus grand ou égal à B.

Indice : A et B sont donc en fait deux nombres écrits chacun sur 1 bit. Listez les cas possibles et déduisez-en le circuit correspondant.

5.b) Comparateur 2-bit

Imaginez ensuite un circuit avec quatre variables d'entrée A_0, A_1, B_0 et B_1 et donc la sortie vaut vrai si et seulement si $\underline{A_1A_0}_2 \geq \underline{B_1B_0}_2$

Indice : Vous pouvez commencer par tracer un circuit qui renvoie vrai si et seulement si $\underline{A_1A_0}_2 = \underline{B_1B_0}_2$ (c'est-à-dire un test d'égalité entre deux nombres écrits sur 2 bits)

5.c) Comparateur n-bit

Pour finir, déduisez en comment comparer deux nombres écrits sur 32 bits ou 64 bits (inutile cependant de réaliser le schéma entier dans *Logic Friday*).

VI. Multiplexeur et démultiplexeur

6.a) Multiplexeur

Un multiplexeur 4x2 est un circuit avec

- 4 entrées de données, notées d_0, d_1, d_2 et d_3
- 2 entrées de sélection, notées i_0 et i_1
- Une sortie s

Les valeurs de vérité des entrées de sélection définissent quelle est l'entrée de données dont la valeur de vérité est envoyée sur la sortie :

$$\text{Si } \underline{i_1i_0}_2 = k \quad \text{alors} \quad s = d_k$$

Imaginez et tracez le circuit correspondant.

Indice : Inspirez-vous du circuit réalisé au 3.b)

6.b) Démultiplexeur

Alors que le multiplexeur permet de récupérer le contenu d'une variable stockée en mémoire, le démultiplexeur permet de modifier le contenu d'une variable en mémoire.

Un tel circuit possède ainsi :

- 2 entrées de sélection, notées i_0 et i_1
- Une entrée valeur v
- 4 sorties, notées s_0 , s_1 , s_2 et s_3

Les valeurs de vérité des entrées de sélection définissent quelle est la sortie sur laquelle sera envoyé l'entrée valeur :

$$\text{Si } \underline{i_1 i_0}_2 = k \quad \text{alors} \quad s_k = v$$

Imaginez et tracez le circuit correspondant.

Indice : Inspirez-vous du circuit réalisé au 3.c)

VII. Affichage analogique**

Essayez de concevoir un circuit booléen composée de :

- 4 entrées, notées i_0 , i_1 , i_2 et i_3
- 7 sorties, correspondant aux 7 segments d'un affichage analogique

et tel que si $\underline{i_3 i_2 i_1 i_0}_2 = k$, alors le chiffre k est affiché.



Défi supplémentaire*** : Essayez de réaliser ce circuit en un minimum de portes.