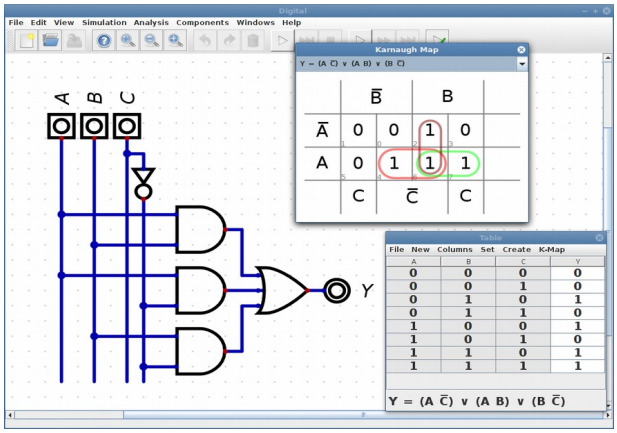
David DEFOUR
Kenelm LOUETSI

TP N°1
1: Introduction aux circuits logiques
Consignes

Outil

Nous utiliserons le logiciel libre ***Digital.*** Il est téléchargeable à :
https://github.com/hneemann/Digital dans la rubrique *Releases*.

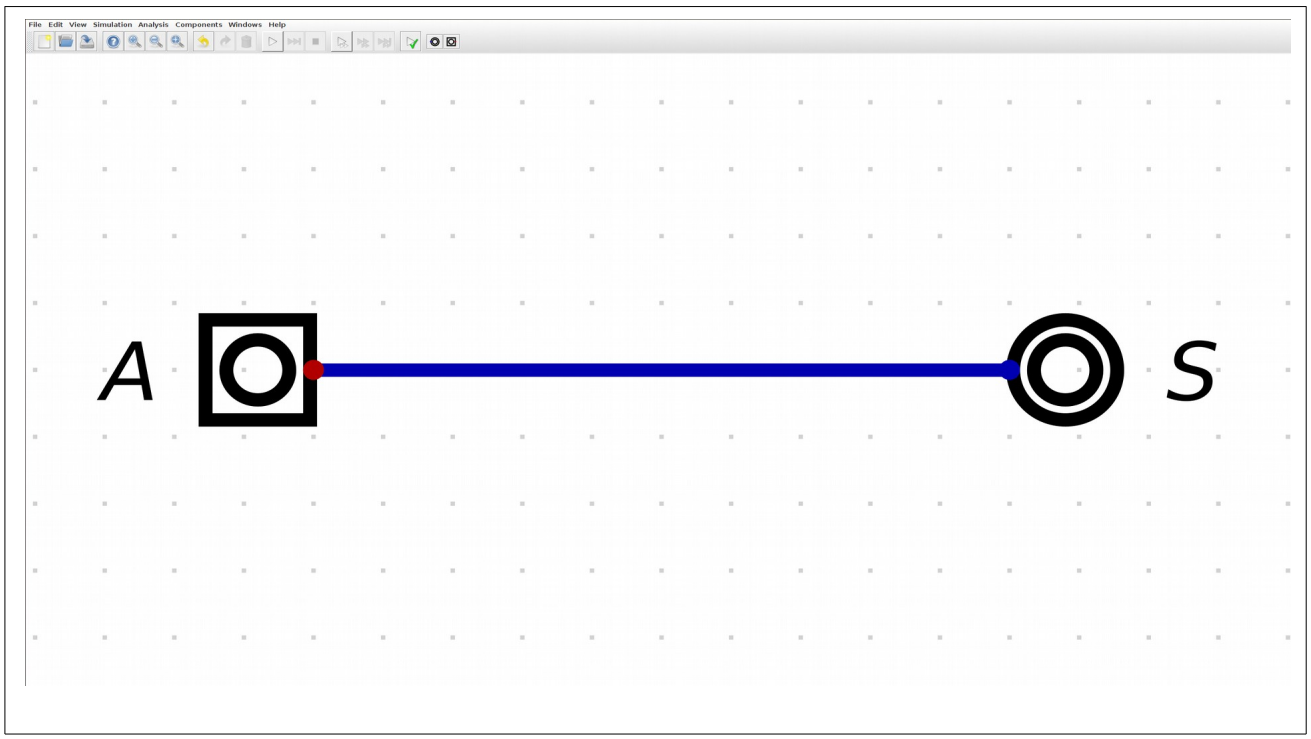


*(les questions sont à compléter sur la «**Fiche réponse**» correspondante)*

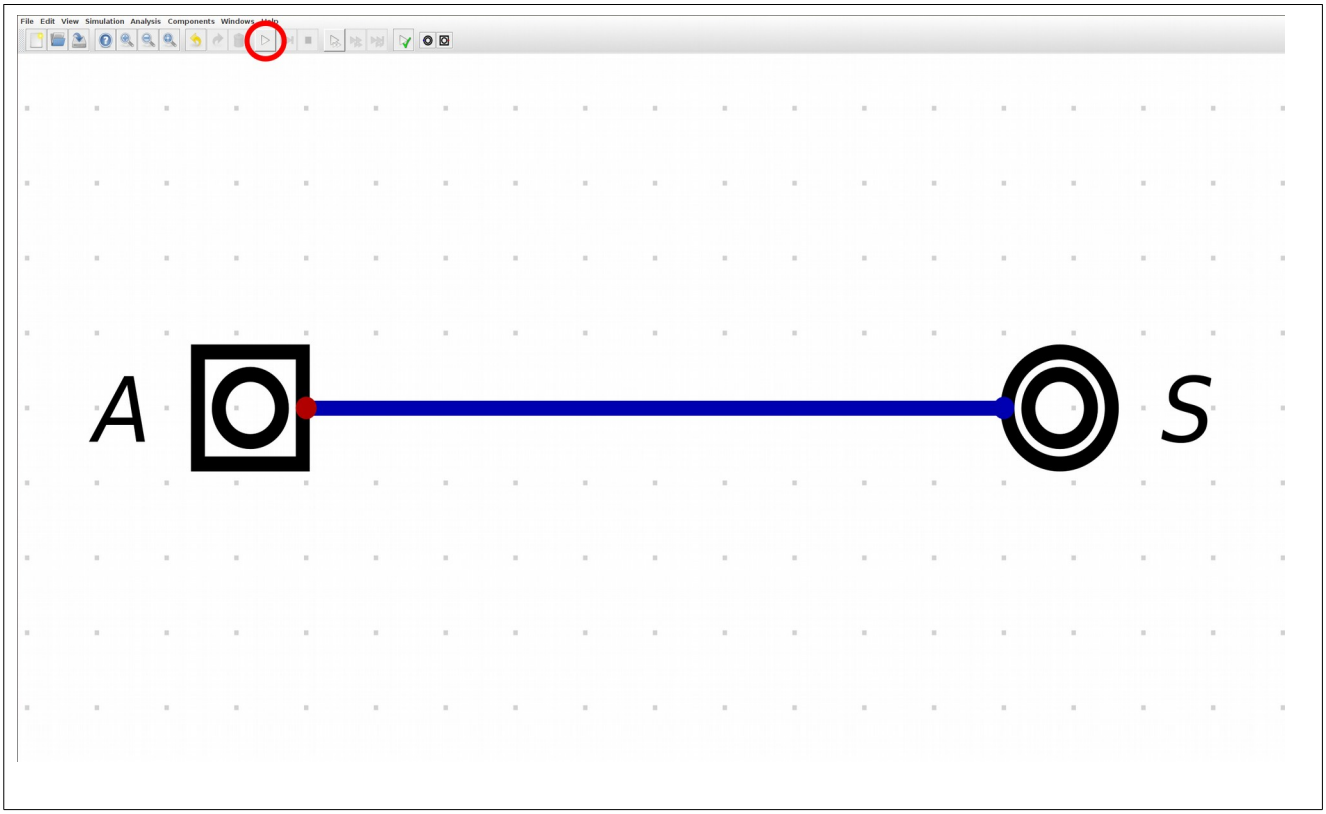
Exercice 1 : Prise en main

1.1

Nous allons tout d'abord faire un simple « interrupteur ».
Pour cela, insérez une entrée et une sortie avec : *Composants > E/S*
Ensuite, reliez les avec le clic gauche et étiquetez l'entrée 'A ' et la sortie 'S' avec le clic droit :



Pour rendre « l'interrupteur » interactif, il faut démarrer une simulation :



Vous pouvez à présent interagir avec les éléments du circuit en cliquant dessus.
Par défaut, ceux-ci sont inactifs, ce qui corresponds à '0'.

Que se passe-t-il si vous faites un clic gauche sur 'A' ?

1.2

On veut à présent ajouter une second interrupteur au circuit, afin que la sortie soit positive dans le cas où uniquement une seule des deux entrées soit positive.
Ajoutez une seconde entrée 'B' et la ou les portes logiques pertinentes au circuit, puis relancez la simulation.
(les portes logiques se trouvent dans : Composants > Logique)

Donnez l'expression du circuit crée.

1.3

Ouvrez le fichier *TP1 > circuits > circuit_1.dig* dans *Digital* et dessinez le circuit implémentant l'expression précédente.

1.4

Donnez la table de vérité du circuit implémenté.
(vous pouvez automatiquement générer la table de vérité de votre circuit via Analyse > Analyse)

1.5

Générez le tableau de Karnaugh du circuit implémenté.
(vous pouvez automatiquement générer le tableau de Karnaugh via Analyse > Analyse > Table-K)

1.6

Quelle fonction logique le circuit dessiné implémente-t-il ?
Une fois le circuit validé, enregistrez et fermez le fichier *circuit_1.dig*

David DEFOUR
Kenelm LOUETSI

Exercice 2 : Additionneur

On veut désormais construire pouvant additionner deux nombres sur 4-bits avec retenue.
Cependant, même un circuit aussi « simple » comporte plusieurs dizaines de portes logiques.
Nous allons donc simplifier le problème en le découpant en plus petits sous-problèmes.

A) Half-adder

2.A.1

Pour commencer, nous allons créer un circuit permettant d'additionner deux nombres de 1-bit sans la retenue.
Ce circuit est a créer dans le fichier *TP1 > circuits > half-adder_1bit.dig*

Quelle porte logique élémentaire permet d'implémenter le circuit ?

2.A.2

On veut pouvoir savoir si une addition de deux nombres à 1 bit génère une retenue.
Il faut donc ajouter une sortie 'Cout' *(Carry out)* et une porte logique élémentaire au circuit précédemment crée.

Dessinez le circuit permettant l'addition de deux nombres de 1-bit avec retenue sortante.

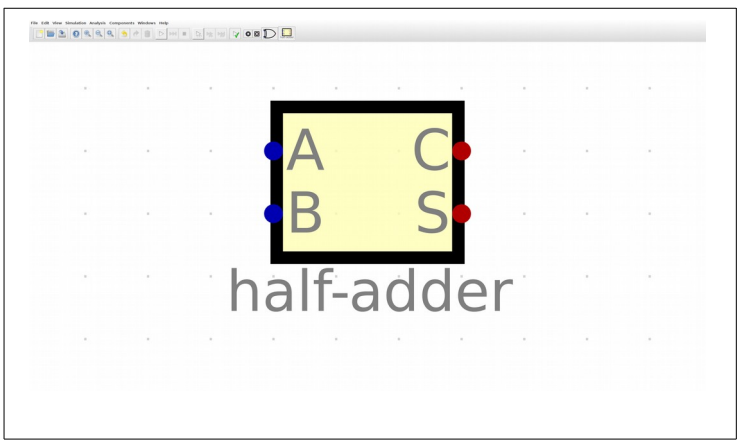
Une fois le circuit validé, enregistrez et fermez le fichier *half-adder_1bit.dig*

B) Full-Adder

2.B.1

Avec notre « half-adder », nous pouvons additionner deux nombre de 1 bit, en connaître le résultat et savoir si il y a une retenue sortante.
Cependant, pour pouvoir additionner des nombres de plus de 1 bit, il faut pouvoir propage les retenues.
Nous allons donc créer un « full-adder » 1-bit, qui prendra trois entrées :'A' et 'B' et 'Cin' (Carry in) et deux sorties 'S' et 'Cout'.
Il est possible de ré-utiliser les circuits déjà enregistrés.
Pour cela, ouvrez le fichier *TP1 > circuits > full-adder_1bit.dig*
Vous devriez à présent pouvoir placer « half-adder » via :

Composants > Personnalisé > half-adder



À l'aide du « half-adder », dessinez le circuit du « full-adder ».

2.B.2

Afin de valider ce circuit, nous allons créer un test avec : *Composants > Divers > Test*
Pour le configurer celui-ci, faites un clic droit dessus allez dans *Éditer*.

Cin	A	B	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

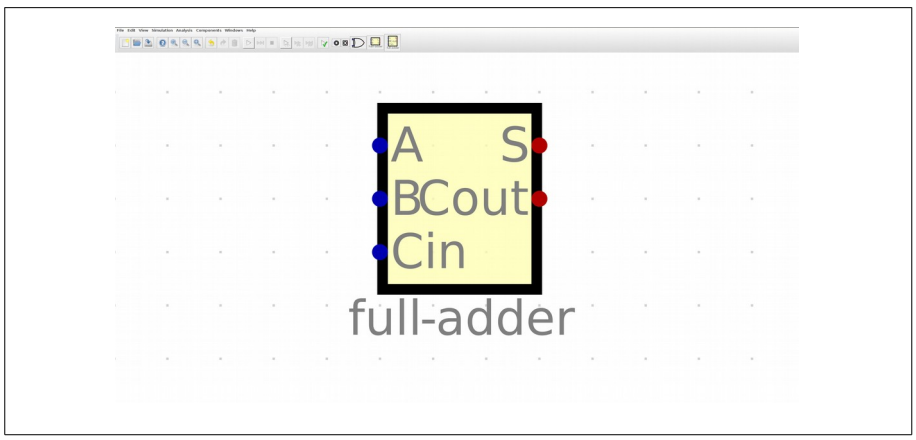
Ajoutez la table de vérité ci-dessus au test et validez votre circuit.

Une fois le circuit validé, enregistrez et fermez le fichier *full-adder_1bit.dig*

C) 4-bits Adder

2.C.1

Nous pouvons désormais dessiner l'additionneur 4-bits en assemblant quatre « full-adder » ensemble.
Celui-ci prends en entrée deux nombres entiers non-signés de 4-bits ('A') et ('B'), une retenue de 1-bit ('Cin') et renvoie en sortie le résultat ('S') sur 4-bits et la retenue ('Cout').
A, B et S se décomposent respectivement en ('A3', 'A3', 'A3', 'A3'), ('B3', 'B2', 'B1', 'B0') et ('S3', 'S2', 'S1', 'S0').



Ouvrez le fichier *TP1 > circuits > full-adder_4bits.dig* et dessinez le circuit de l'additionneur 4-bits.

2.C.2
Indiquez combien existe-t-il de combinaisons possibles pour l'addition de deux nombres entiers non-signés de 4-bits avec 1-bit de retenue.

2.C.3
On voudrait pouvoir valider ce circuit, pour cela on peut se servir de sa table de vérité.
Pour éviter de passer un temps considérable à écrire la table de vérité, on va automatiser le test avec un script.

```
Cin A3 A2 A1 A0 B3 B2 B1 B0 Cout S3 S2 S1 S0

loop(A,16)
  loop(B,16)
    0 bits(4,A)  bits(4,B)  bits(5,A+B)
    1 bits(4,A)  bits(4,B)  bits(5,A+B+1)
  end loop
end loop
```

Ce script va tester toute les combinaisons d'entrées possible de l'additionneur 4-bits.

Ajoutez un test, ajoutez lui le script ci-dessus et validez votre circuit.

2.C.4
Générez le chronogramme du test de l'additionneur 4-bits.
(vous pouvez automatiquement générer un chronogramme à partir de vos résultats de tests)

Une fois le circuit validé, enregistrez et fermez le fichier *full-adder_4bits.dig*

Exercice 3 : Complément à 2

3.1
On veux pouvoir obtenir le complément à 2 d'un nombre entier non-signé sur 4-bits.

Rappelez les étapes de la conversion d'un entier en complément à 2.

3.2
Ouvrez le fichier *TP1 > circuits > 2s-complement_4bits.dig* et dessinez le circuit du convertisseur nombre entier non-signé vers complément à 2.