TP Architecture

M. Tellene

1 Création de l'environnement de travail

Avant de commencer le TP, vous allez créer votre environnement de travail. Pour ce faire vous allez dans votre dossir personnel et créez un dossier que vous nommerez « **TP_architecture** ». C'est dans ce dossier que vous sauvegarderez les exercices de ce TP.

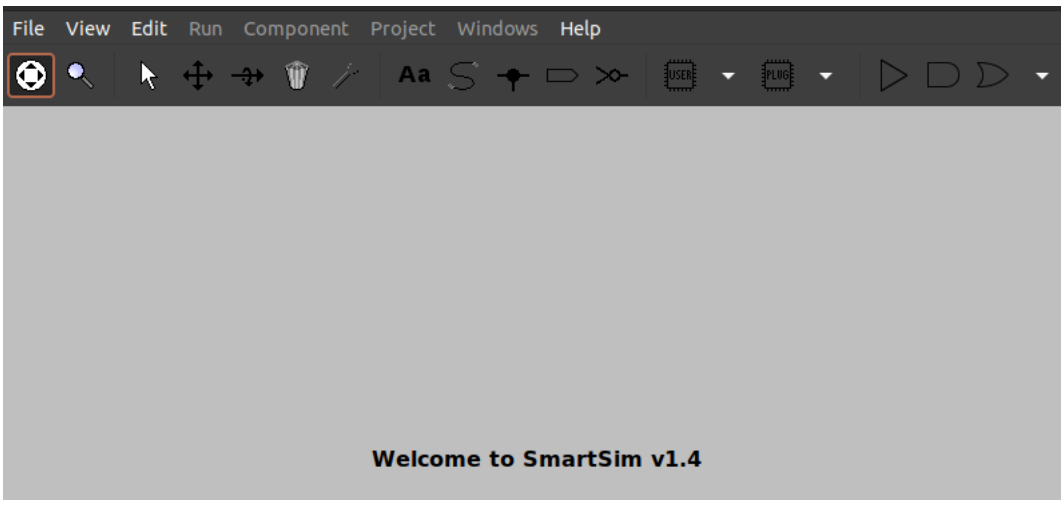
2 Introduction

Les circuits d'ordinateurs manipulent **uniquement des 0 et des 1** qui en interne sont représentés par des tensions électriques.

- 0 est représenté par une tension basse : 0V
- 1 est représenté par une tension haute : +xV

3 Le simulateur *SmartSim*

SmartSim est un logiciel permettant de représenter des circuits logiques qui peuvent notamment prévoir l'état d'une sortie pour une configuration d'entrées données. En démarrant *SmartSim* (*Menu démarrer → Education → SmartSim*), vous devriez avoir une interface comme ceci :



4 A vous de jouer !

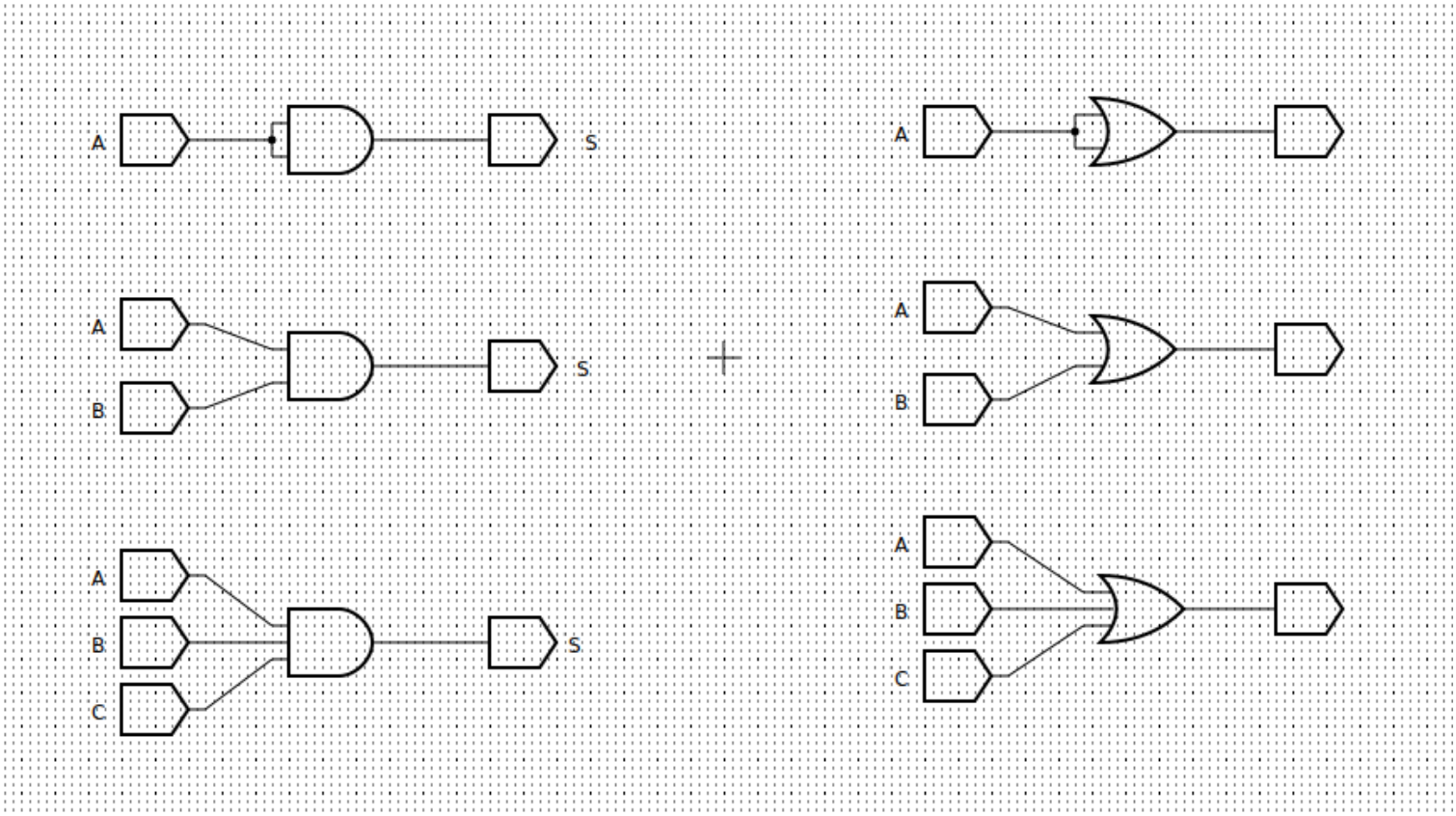
Une fois sur le logiciel *SmartSim*, faites File → New Project puis faites File → Save Project As puis sauvegarder dans votre dossier « TP_architecture ». Vous appellerez ce fichier « exercices_tp ». Une fois fait, refaites Files → New Component, si tout c'est bien passé, le fond de la partie basse à dû changer en un quadrillage à points.

4.1 Prise en main

EXERCICE 1

Ce premier exercice a pour but de vous faire prendre en main le logiciel. **Vous allez commencer par sauvegarder le composant actuel : File → Save Component As → Aller dans « exercice_tp → Appeler ce composant « prise_en_main ».**

Pour ce premier exercice, réaliser les circuits suivants :



1. Exprimer le comportement de chacun de ces circuits à l'aide de tables de vérité

Un peu d'aide :

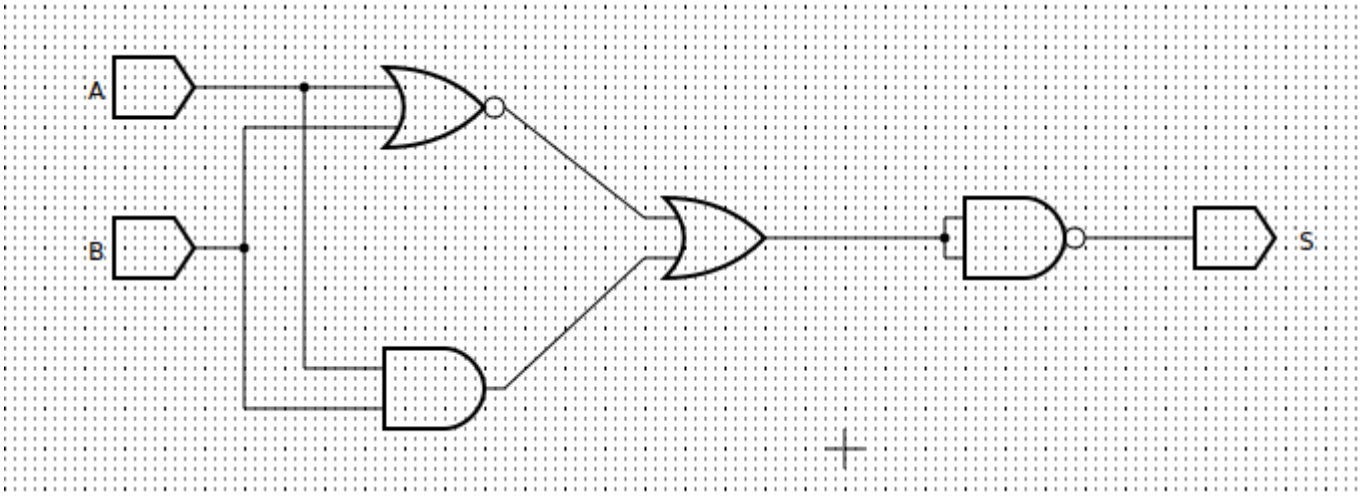
- Passer votre souris sur les éléments pour voir leurs noms ainsi que le mode d'utilisation
- les entrées sont l'élément *Toggle*
- les sorties sont l'élément *Reader*
- les labels sont l'élément *Annotate*
- les fils sont l'élément *Wires*
- **pour tester vos circuits, il faut faire une fois Component → Set As Root, puis vous pouvez faire Run → Run**
- pour créer une porte logique avec un nombre de pins différent de 2, il faut cliquer sur *Adjust* → cliquer sur la porte à modifier → entrer le nombre de pins → Apply
- quand votre circuit est lancé, cliquez sur les entrées pour les faire passer de 0 à 1 et inversement

4.2 Les exercices

EXERCICE 2

Pour cet exercice, il faudra créer un nouveau fichier : simplification_expression.

1. Reproduire le circuit suivant :



2. En utilisant une table de vérité, exprimer le comportement à la sortie de chacune des portes logiques du circuit en fonction des entrées `A` et `B`, ainsi que l'état de la sortie `S`
3. A l'aide du site suivant : https://fr.wikipedia.org/wiki/Alg%C3%A8bre_de_Boole_(logique) (aller dans la partie « Fonctions logiques »), déterminer quel opérateur logique a été construit

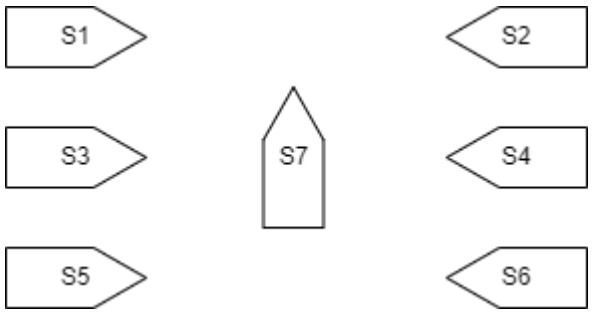
— pour créer un `NOR` (`NOT` + `OR`), il vous suffit de mettre une porte `OR` → cliquer sur *Invert* → cliquer sur le pin de sortie de la porte `OR`, un rond devrait apparaître. Même chose pour un `NAND` (`NOT` + `AND`) mais en remplaçant la porte `OR` par une porte `AND`

EXERCICE 3

Pour cet exercice, il faudra créer un nouveau fichier : dé

Le but de cet exercice est de fabriquer un dé.

1. Sur combien de bits peut-on coder toutes les valeurs d'un dé? Pourquoi?
2. Sur votre fenêtre de travail *SmartSim*, disposer 7 sorties de la façon suivante :



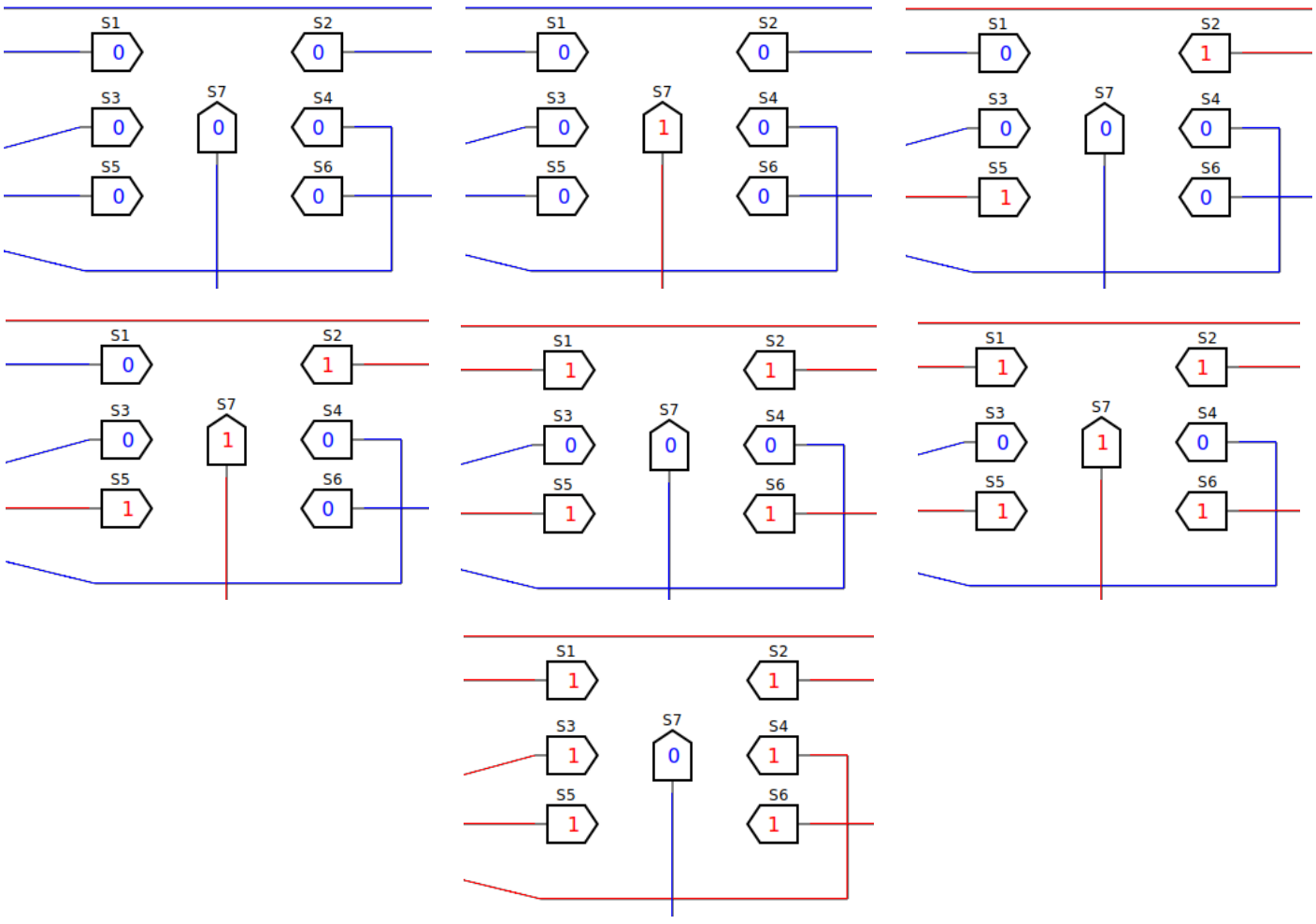
3. Sur votre feuille réponse, faire la table de vérité de chaque sortie en fonction de chaque bit, la structure ainsi qu'un morceau de la première ligne sont donnés :

   Rappel et indication :
   — b2 = b1 = b0 = 0 (000_2) correspond à 0
   — b2 = b1 = 0 et b0 = 1 (001_2) correspond à 1

— La sortie 1 est symbolisée par S1 dans la table de vérité suivante

N	b2	b1	b0	S1	S2	...
0	0	0	0	0	0	
1	0	0	1	...	...	
2	0	1	0	...	...	
3	...	...	...	...	...	
4	...	...	...	...	...	
5	...	...	...	...	...	
6	...	...	...	...	...	
7	...	...	...	...	...	

4. Créer, sur *SmartiSim*, le circuit permettant d'avoir le résultat suivant : (afin de simplifier, nous dirons qu'il est impossible de faire 7, pas besoin de gérer ce cas)



EXERCICE 4

Pour cet exercice, il faudra créer un nouveau fichier : 7_segment

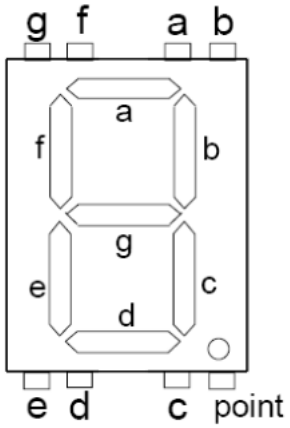
A l'image de l'exercice précédent, une valeur numérique peut être représentée sous forme de segment permettant de distinguer les différentes valeurs. Pour cet exercice, nous allons créer un afficheur 7-segment.

Pour rappel (ou information) un afficheur 7-segment permet d'afficher les chiffre de 1 à 9 dans la forme suivante :

1. Sur combien de bits peut-on coder toutes les valeurs d'un 7-segment ? Pourquoi ?

2. En suivant l'identification des LEDs suivante, dresser la table de vérité de notre afficheur 7-segment

   Important : Un afficheur 7-segment n'affiche que 10 chiffres, les sorties 10 ≤ N ≤ 15 seront mises à 1 pour tous les segments. Cela ne changera en rien le comportement de notre afficheur.



3. **Faites sur papier**, le circuit logique permettant de faire fonctionner le 7-segment