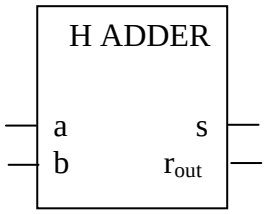
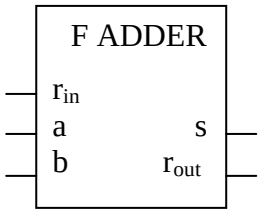
TP 1. ADOintro - Codage

Exercice 1



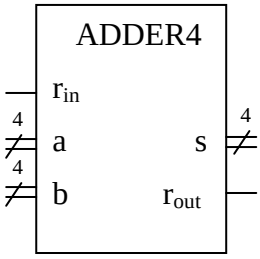
a) Réaliser sous CircuitMaker une macro (composant réutilisable) un composant demi-additionneur 1 bit : **HalfAdder**.
b) Faire un test de la macro avec un fichier .ckt utilisant la macro **HalfAdder**.
c) Simuler votre test (scénarios de test avec des *logic displays*).

Exercice 2



a) Réaliser sous CircuitMaker un additionneur complet 1 bit : **FullAdder** qui réutilise votre demi-additionneur.
b) Faire un test.
c) Simuler votre test (scénarios de test avec des *logic displays*).

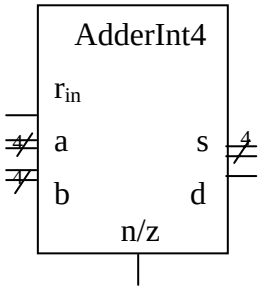
Exercice 3



a) Réaliser sous CircuitMaker un additionneur 4 bits : **Adder4** qui réutilise 4 additionneurs complets. Ce composant a une retenue entrante et une retenue sortante.
b) Faire un test.
c) Simuler votre test (scénarios de test avec des *hex displays*).

Devices → Digital animated → Displays → Hex display

Exercice 4 (Facultatif)



a) Réaliser sous CircuitMaker un additionneur sur des nombres sur 4 bits naturels ou relatifs : **AdderInt4** qui réutilise l'additionneur 4 bits précédent. On garde la retenue entrante et on rajoute un signal n/z qui permet de sélectionner le type des opérandes (1 pour les naturels, 0 pour les relatifs). En sortie, on remplace la retenue sortante par un signal de dépassement de capacité.
b) Faire un test.
c) Simuler votre test (scénarios de test avec des *hex displays*).

Compte-rendu

Faire un compte-rendu du TP1 incluant les schémas de vos composants, les sources CircuitMaker et les scénarios de simulation commentés.

Facultatif : Optimisation du calcul de retenue (retenue par anticipation *versus* propagation de retenue)

Le temps de traversée de l'additionneur 4 bits est dû en grande partie à la propagation de la retenue entre les différents additionneurs complets (exemple 0001 + 1111). Nous allons étudier une solution qui permet d'optimiser ce calcul. On définit les fonctions suivantes pour i = 0 à 3 (0 poids faible, 3 poids fort).

- génération de retenue : $g_i = a_i.b_i$
- propagation de retenue : $p_i = a_i \text{ xor } b_i$
- calcul de la retenue : $c_i = g_i + p_i.c_{i-1}$
- calcul de la somme : $s_i = p_i \text{ xor } c_{i-1}$

a) expliquer ces formules (on pourra remarquer que (g_i,p_i) constituent la sortie d'un ½ additionneur) ;
b) donner les formules de s_0, c_0, s_1, c_1, s_2, c_2, s_3, c_3 en fonction de g_0, p_0, g_1, p_1, g_2, p_2, g_3, p_3 et c_e (la retenue entrante) ;
c) développer ces formules pour obtenir un temps de propagation minimal ;
d) en déduire le schéma du nouvel additionneur 4 bits à partir de 4 ½ additionneurs
e) réaliser le circuit sous CircuitMaker
f) vérifier le gain dans le temps de traversée de l'additionneur 4 bits ; on pourra utiliser des portes OR et AND à un nombre quelconque d'entrées.

Facultatif : VHDL *versus* CircuitMaker

Reprendre les différentes questions du TP avec une solution logicielle (VHDL) plutôt que matérielle (CircuitMaker).