Printemps 2022
RAKOTOVAO Jonathan
VOYLES Evan

Circuits logiques

TP 1

Objectifs

- ✯ Se familiariser avec logisim
- ✯ Modéliser une formule logique
- ✯ Simuler un circuit logique

1 Combinatorial Circuit

a	b	c	$(b \cdot \bar{c})$	s
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	0	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	1	0	1

2 Circuit 7 segments

- ✯ Ecrire la table de vérité du sous-circuit `Bit1`

Le `Bit1` doit être allumé pour les chiffres (en décimal) qui appartient à l'ensemble $\{0, 1, 3, 4, 5, 6, 7, 8, 9\}$. Sa table de vérité est donc :

Chiffre	a	b	c	d	s
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
E	1	x	x	x	0

Pour construire le sous-circuit `Bit4`, on procède d'une manière analogue du `Bit4`. On doit tout d'abord établir pour quel chiffres doit-on allumer la position $b4$ dans le circuit 7 segments. Une analyse nous révèle que `Bit4` doit être allumé pour les chiffres $\{0, 4, 5, 6, 8, 9\}$.

Chiffre	a	b	c	d	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
E	1	x	x	x	1

A partir de cette table de vérité on a pu générer automatiquement le circuit suivant

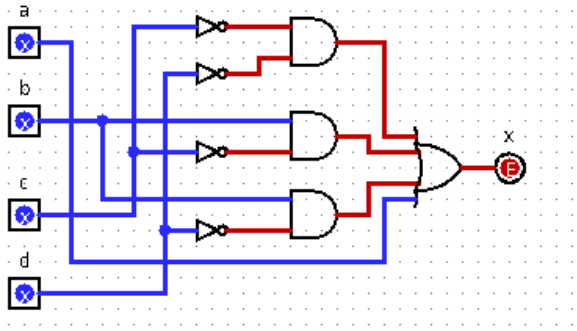

FIGURE 1 – Circuit de `Bit4`. Il est encodé par la fonction booléenne $s = a + b\bar{d} + b\bar{c} + \bar{c}\bar{d}$

Le bug qu'on a trouvé c'est que `Bit6` est allumé incorrectement quand on veut visualiser le chiffre 7. Alors pour résoudre le problème on modifie la table de vérité de `Bit6` pour que $\texttt{Bit6}(0, 1, 1, 1) = 0$. Enfin on génère le circuit selon la table modifié et on trouve que le bug est résolu.

3 Conception d'une unité arithmétique et logique (ALU)

L'opération d'addition en binaire est presque pareil de celui en décimal, pourtant en binaire on a que 2 chiffres, $\{0, 1\}$, pour représenter les nombres. De même, il s'agit d'additioner deux nombres chiffres par chiffres, en gardant un chiffre de retenue quand on dépasse le maximum chiffre qu'on a dans notre base.

$A[0]$	$B[0]$	Cin	$O[0]$	V	cout
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	1	1

Pour transformer un nombre en sa version négative selon l'encodage de complément à 2, on prend le complément à 1 - c'est-à-dire l'inversion de tous les bits d'un nombre - et après on ajoute 1 à ce nombre inversé. Si on dédie un bit d'entré pour choisir l'opération à executer (1 pour la soustraction, 0 pour l'addition), on peut utiliser une porte XOR avec le bit d'opération et chaque bit de notre entier de 32bits pour inverser les bits quand le bit d'opération est allumé. Pour visualiser cela on écrit la table de vérité d'une XOR.

a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

Si on se concentre sur la moitié en bas où a est vrai, on remarque que la value de c est tout simplement $\bar{b}$. Alors on donne "à manger" tous les 32 bits dans 32 portes XOR pour avoir le complément à 1 du nombre et ensuite on donne en entrée ce nombre plus 1 pour faire une addition, où le 'plus 1' remplit le vide du bit de retenue en entrée du 32-bit adder. Le circuit est bouclé, on a conçu une ALU.