TP 2

LOGISIM- Circuits combinatoires part. 2

Fichiers fournis : `tp2_aluetu.circ`, `tp2_pgcdetu.circ`

2.1 Quelques éléments du processeur LC-3

Dans la suite du cours, nous allons construire un petit processeur pédagogique, le LC-3. Nous prenons de l'avance dans ce TP en construisant quelques sous-circuits que nous assemblerons ensemble dans un prochain TP.

EXERCICE 1 ▶ **ALU LC-3**
Récupérez sur la page web du cours le fichier `tp2_aluetu.circ` et testez-le pour savoir ce qu'il fait. Remplissez les cases vides du tableau suivant avec des formules dépendant des entrées `Input1`, `Input2` et `Cst` :

e_2/`UseCst`	0	1
(00)		
(01)		
(10)		
(11)		

EXERCICE 2 ▶ **NZP LC-3**
Dans un nouvel onglet du fichier précédent (nommé `NZP`), créez un circuit qui prend une entrée 16 bits nommée RES considérée en complément à 2 sur 16 bits, et qui en sortie a un "Pin" 3 bits nommé NZP. Le bit de poids faible (P) est égal à 1 ssi $RES > 0$, le bit du milieu (Z) est égal à 1 ssi $RES = 0$, et le bit de poids fort est à 1 ssi $RES < 0$. Testez bien.

EXERCICE 3 ▶ **Extensions de signe**
D'après un des exercices de TD, l'extension de signe en complément à 2 se fait en dupliquant le bit de poids fort autant de fois que nécessaire. Créez dans un même fichier deux onglets différents :

— Dans un onglet appelé `Ext8vers16`, construisez un sous-circuit pour l'extension de signe d'un entier codé en complément à 2 sur 8 bits vers 16 bits. Testez le sous-circuit dans un onglet `Brouillon`.

— Comparez votre sous-circuit `Ext8vers16` avec le composant `BitExtender` de la librairie (dans Wiring).

2.2 Circuits combinatoires

EXERCICE 4 ▶ **Additionneurs à propagation simple de retenue**
Dans un nouveau fichier :

— Réalisez l'additionneur 1 bit à retenue du cours et testez-le.

— Regardez dans la documentation comment fonctionne l'encapsulation (`Subcircuits`). Nommez l'additionneur 1 bit "FA1" et utilisez le pour réaliser un additionneur 4 bits.

— Utilisez l'additionneur 8 bits de la librairie (`Arithmetic->Adder`) avec des "constantes" (`Wiring->Constant`) en entrée de l'addition et un afficheur (`Probe`) 8 bits en sortie. On vérifiera que $(80)_{16} + (8C)_{16} = (00001100)_2$ (et 1 de retenue).

— En utilisant cet additionneur 8 bits (et les multiplexeurs de la librairie), réalisez un additionneur-soustracteur 8 bits, qui calcule $a - b$ ou $a + b$ suivant la valeur d'un bit de contrôle c. Vous n'avez pas le droit de dupliquer l'additionneur (vous pouvez vous reporter à l'exercice correspondant du cahier de TD).

— Réalisez une **ALU 8 bits** capable de faire une addition, une soustraction et un test d'égalité. L'opération sera choisie avec un signal qui vaut 00 pour une addition, 01 pour une soustraction et 10 pour un test d'égalité. On remarquera que c'est une modification mineure du circuit précédent.

EXERCICE 5 ▶ **Additionneur à propagation rapide de retenue - sélection de retenue**

L'inconvénient des additionneurs 8 bits en cascade est que chaque additionneur 1 bit doit attendre que sa retenue entrante soit disponible pour réaliser l'opération. Un additionneur 8 bits a donc un temps de traversée égal à 8 fois le temps de traversée d'un additionneur 1 bit. Un additionneur à sélection de retenue (*carry select*) peut être construit en utilisant le temps de traversée d'un additionneur 4 bits (utilisé pour additionner les 2×4 bits de poids faible) pour précalculer les deux résultats possibles de l'addition des 2×4 bits de poids forts (l'un avec une retenue entrante égale à 1, l'autre avec une retenue entrante nulle). Un multiplexeur est utilisé pour sélectionner le bon résultat lorsque la retenue entrante est finalement connue.
Réalisez un tel additionneur en utilisant l'additionneurs 4 bits de la question précédente.

EXERCICE 6 ▶ **Une calculette à PGCD**

Nous allons construire un circuit qui réalise le calcul du PGCD pour les entiers positifs (8 bits). La figure 2.1 vous fournit une définition et un programme C pour ce calcul.

> "En arithmétique élémentaire, le plus grand commun diviseur, abrégé en général PGCD, de deux nombres entiers naturels non nuls est le plus grand entier qui divise simultanément ces deux entiers. Par exemple le PGCD de 20 et 30 est 10. En effet, leurs diviseurs communs sont 1, 2, 5 et 10."

Listing 2.1 – 'Afficherec.asm'

```
int gcd(int x, int y){
  while (x != y){
    if (x<y) y=y-x;
    else     x=x-y;
  return x;
}
```

FIGURE 2.1 – Documentation pour le PGCD (Wikipédia)

— Quel est le pgcd de 12 et 8?

— Expliquez la différence entre cet algorithme et celui que vous connaissez pour calculer le pgcd?

Pour vous simplifier la vie, nous vous fournissons un circuit (figure 2.2) qui possède déjà les composants de données et de calcul. Nous n'aurez plus qu'à ajouter les composants de contrôle.

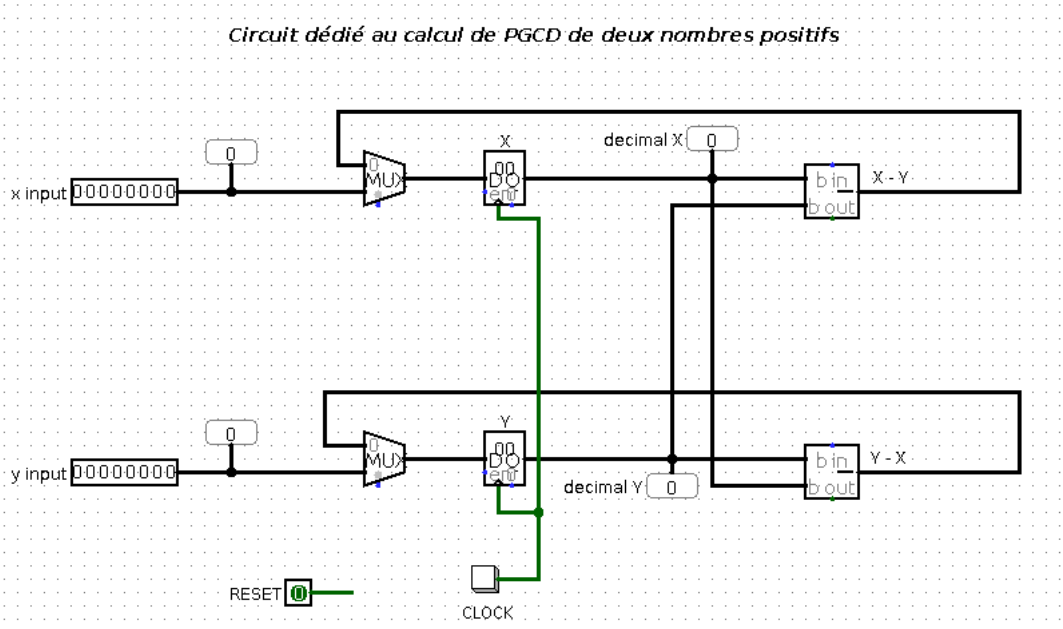

FIGURE 2.2 – Le circuit fourni

— Sur la page web du cours, téléchargez le circuit `tp2_pgcdetu.circ`.

— Observez le circuit : les entrées de gauche seront utilisées pour rentrer les valeurs initiales pour x et y (en binaire). Des sondes (`Probe`) décimales ont été ajoutées pour pouvoir lire ces valeurs en base 10 (ainsi que les valeurs en sortie des registres).

— Remarquez bien qu'un "tick" d'horloge effectue 1 étape dans le calcul, *le calcul n'est pas instantané!*

— Rajoutez les composants de contrôle : quand les entrées sont disponibles, l'entrée (Pin) `reset=1` doit permettre d'initialiser les registres X et Y avec ces valeurs (au premier appui sur `Clock`). Ensuite, on remettra l'entrée `reset` à 0. Puis chaque appui du bouton `Clock` cause l'exécution d'une étape de l'algorithme. Une fois que le PGCD est trouvé, plus aucun changement ne doit arriver. *On pourra judicieusement se poser les deux questions suivantes : quelles sont les conditions de sélection de l'entrée 1 du multiplexeur du haut (resp. du bas)? Quelles sont les conditions d'écriture de chacun des registres (entrée* `Enable`*)?*

— Vérifiez avec $x = 42$ et $y = 56$. Le PGCD trouvé est?

EXERCICE 7 ► **Additionneur à propagation rapide de retenue - anticipation de retenue**

Nous avons vu précédemment l'additionneur à sélection de retenue. Ce circuit est rapide mais présente deux inconvénients :

— Le résultat d'un des deux additionneurs des bits de poids forts n'est pas utilisé,

— Deux multiplexeurs sont nécessaires pour le résultat de l'addition des bits de poids forts et pour la retenue sortante.

La surface utilisée est donc plus grande, d'où une plus grande consommation électrique et plus de chaleur produite.
Dans cet exercice nous allons réaliser un additionneur 8 bits à anticipation de retenue.
En vous inspirant du cours,

— Reprenez un additionneur complet 1 bit de l'exercice sur l'additionneur à propagation simple de retenue et ajoutez les sorties de génération et de propagation de retenue ;

— Créez un additionneur 4 bits avec circuit d'anticipation de retenue ;

— Créez un additionneur 8 bits couplant deux additionneurs 4 bits avec circuits d'anticipation de retenue, la retenue entrante de l'additionneur des bits de poids forts étant la sortie du circuit anticipateur de retenue de l'additionneur des bits de poids faibles ;

— Comparez les temps de passage avec l'additionneur 8 bits à sélection de retenue.