julien.braine@ens-lyon.fr,mateusz.skomra@ens-lyon.fr, florent.de-dinechin@insa-lyon.fr

ASR1, TD3 : circuits combinatoires

Ce TP et les suivants utilisent Digital. Commencez par l'installer :
- Télécharger `https://github.com/hneemann/Digital/releases/latest/download/Digital.zip` (on le trouve sur le Ternet par la requête "digital logisim" car Digital est un clone de Logisim)
- le décompresser quelquepart
- aller dans le répertoire Digital ainsi créé, et taper `java -jar Digital.jar`

Exercice 1: Introduction à Digital :

Au premier lancement, un tutoriel vous est proposé : déroulez-le. Au cas ou mais j'en doute, la solution est au début de l'aide de Digital (Help->Documentation). Il est d'ailleurs recommandé de vous référer fréquemment à l'aide de Digital, qui est très bien faite !

Commencez ensuite cet exercice dans un nouveau fichier.

1.1) Réalisez un multiplexer 2 bits vers 1 bits, et testez le. Vous utiliserez pour cela des portes `AND` à 2 entrées, en inversant certaines entrées, ainsi qu'une porte `OR`. Sauvegardez votre multiplexeur 2 vers 1 dans un fichier que vous appellerez `Mux21`.

1.2) En utilisant des portes `AND` à 3 entrées, réalisez un autre circuit `Dec38`, un décodeur 3 bits vers 8.

1.3) Faites une copie de votre circuit `Dec38`, puis complétez-là en un multiplexeur 8 bits vers 1 bit. Une nouveauté : l'entrée de sélection devra être un vecteur de 3 bits (bouton droit, *data bits*). Il faudra utiliser le composant `Splitter` de la catégorie `Wires`.

1.4) Construisez un nouveau circuit multiplexeur de 8 bits vers 1, en assemblant des `Mux21` comme dans le poly. Pour cela il faudra aller lire la section 1.4 *hierarchical design* de la doc de Digital.

Exercice 2: Une unité arithmétique et logique :

Commencez cet exercice dans un nouveau fichier, disons `alu.circ`. Le but est de construire en Digital une petite ALU *(Arithmetic and Logic Unit)* 8 bits, capable d'effectuer la conjonction et la négation bit-à-bit, ainsi que l'addition et la soustraction en complément à deux.

2.1) Rappelez la construction de l'additionneur à partir de cellules *full adder*. Ecrivez la table de vérité du *full adder*.

2.2) Construisez, à base de portes de bases, un sous-circuit `FA` qui implémente le *full adder*, en suivant stupidement la méthodologie automatique vue en cours.

2.3) Créez un sous-circuit `addsub`, contenant un additioneur/soustracteur en complément à deux sur 8 bits. Le circuit :
- prend en entrée deux entiers `a` et `b`, ainsi qu'une entrée `sub`
- produit en sortie `s`, et un indicateur de dépassement de capacité `dep`.

En l'absence de dépassement de capacité, `s` prend pour valeur $a+b$ si $sub = 0$, $a-b$ si $sub = 1$. L'indicateur `dep` prend pour valeur 1 en cas de dépassement de capacité, 0 sinon.

2.4) Réalisez ensuite l'ALU dans un sous-circuit `ALU`. Le circuit :
- prend en entrée deux entiers `a` et `b`, et qu'une entrée de contrôle `c` ;
- produit en sortie `r`, et un indicateur de dépassement de capacité `dep`.

Les valeurs produites dépendent de `c` :
- si $c = (00)$, alors $r = \overline{a}$, et $dep = 0$,
- si $c = (01)$, alors $r = a\&b$, et $dep = 0$,
- si $c = (10)$, alors $r = a+b$ et `dep` indique les dépassements,
- si $c = (11)$, alors $r = a-b$ et `dep` indique les dépassements,

Utilisez votre `addsub`, et les composants fournis par la bibliothèque de Digital (`Splitter`,...).

Vous pouvez aussi utiliser l'outil `synthesize` du menu `Analysis`, pour générer automatiquement de petits circuits combinatoires à partir d'une expression logique (utile pour l'indicateur de dépassement de capacité, même si ce n'est pas indispensable...).

Exercice 3: Retenue anticipée : En cours, vous avez (peut-être) vu comment presque diviser par 2 le délai d'une addition 32 bits. Sinon, retrouvez-le.

3.1) En faisant une récurrence sur cette idée, quelles sont les complexités en temps et en espace (nombre de portes) de l'addition de n bits ? Et pour l'additionneur naïf ?

3.2) Réalisez en Digital un additionneur 8 bits à propagation de retenue anticipée en utilisant des additionneurs 4 bits de la librairie.

3.3) Sur le même principe, réalisez ensuite un additionneur 16 bits.

Exercice 4: Multiplication binaire :

4.1) Posez une multiplication en binaire. Observez qu'on peut faire toutes les multiplications bit à bit en parallèle. Essayez d'en déduire les complexités en temps et en espace de la multiplication binaire.