TP4 : Conception d'une architecture PC/PO : circuit PGCD sur FPGA

Préparation : Relire et comprendre le TD 6 et lire entièrement l'énoncé de ce TP avant la séance.

Le but de ce TP est d'implanter le circuit PGCD vu en TD et de le faire fonctionner sur la carte FPGA. On rappelle l'algorithme utilisé :

```
procedure PGCD is
  A, B: Positive; -- A et B sont codés sur 8 bits
begin
  Get(A); Get(B);
  while A /= B loop
    if A < B then
      B := B - A;
    else
      A := A - B;
    end if;
  end loop;
  Put(B);
end;
```

Récupérer l'archive du TP sur Chamillo et la décompresser à l'endroit de votre choix.

Ex. 1 : Partie opérative

Le fichier `vhd/PO.vhd` contient un squelette de la partie opérative du circuit. Pour le compléter, vous aurez besoin des composants fournis suivants :

- `FD8CE` : un registre 8 bits avec une entrée d'activation synchrone et une entrée de remise à zéro asynchrone,
- `COMPM8` : un comparateur d'entiers sur 8 bits,
- `ADSU8` : un additionneur / soustracteur sur 8 bits.

La documentation des composants `COMPM8` et `ADSU8` est donnée en annexe.

Question 1 Dessiner le schéma de la partie opérative du circuit avec les composants ci-dessus et en utilisant les noms de signaux de PO.vhd.

Question 2 Ouvrir le fichier `vhd/PO.vhd` et le compléter. Vous aurez besoin des informations suivantes :

- Pour construire un multiplexeur en VHDL, on peut utiliser l'affectation concurrente conditionnelle. Par exemple, `S <= E1 when C = '1' else E0`, permet de décrire un multiplexeur à 2 entrées de données `E0` et `E1`, commandé par l'entrée de contrôle `C`. La sortie `S` doit être du même type que les entrées de données.
- En VHDL, les valeurs logiques doivent être mises entre apostrophes (`'0'`, `'1'`).
- Lors d'un `port map`, si certaines sorties ne sont pas utilisées, on pourra les connecter à `open`.

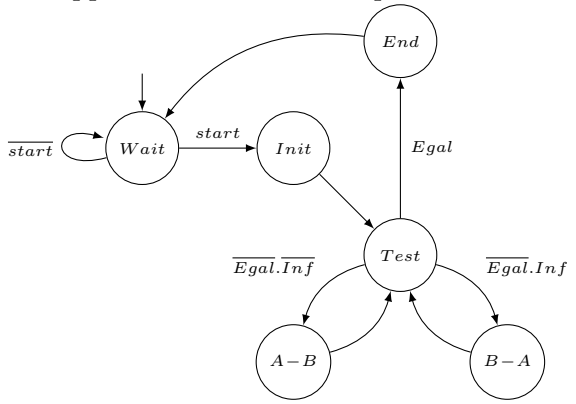
Question 3 Lancer la simulation sur le banc d'essai `vhd/tb_PO.vhd` en utilisant la commande :
`$ make run_simu TOP=PO`
Ajouter les signaux permettant d'observer les valeurs des deux registres.

La simulation doit se terminer par le message *Simulation terminée*. Si des messages d'erreurs apparaissent dans le simulateur, corriger la PO.

Ex. 2 : Partie contrôle

On doit maintenant implanter la partie contrôle du circuit PGCD. Pour cela, on peut choisir d'utiliser un codage compact ou 1-parmi-N pour l'automate. L'avantage du codage 1-parmi-N est qu'on peut déduire les transitions et les sorties directement de l'automate (sans passer par des tableaux de Karnaugh), au prix d'un schéma plus gros (puisqu'il y aura une bascule D par état).

On rappelle l'automate de la partie contrôle du circuit.



	Sorties					
Etat	*GetA*	*LdA*	*GetB*	*LdB*	*SubBA*	*done*
Wait	0	0	0	0	0	0
Init	1	1	1	1	0	0
Test	0	0	0	0	0	0
$A-B$	0	1	0	0	0	0
$B-A$	0	0	0	1	1	0
End	0	0	0	0	0	1

Question 1 Compléter le fichier `vhd/PC.vhd` pour y implanter l'automate suivant le codage un-parmi-N. On utilisera les composants `BDC` (bascule D avec entrée de mise à 0) et `BDS` (bascule D avec entrée de mise à 1) pour représenter respectivement les états et l'état initial.

Ex. 3 : Simulation du bloc PC/PO

Le fichier `vhd/pgcd.vhd` fourni instancie les parties PO et PC déjà réalisées. Ce fichier peut être testé grâce au fichier `vhd/tb_pgcd.vhd` et à la commande :

`$ make run_simu TOP=PGCD`

Question 1 Lancer la simulation en organisant de manière lisible comme vu au TP3 (radix, séparateur, couleur).

Quelles sont les valeurs d'entrées utilisées pour les ports A0 et B0 ? Vérifier que la valeur fournie par votre circuit correspond bien au PGCD de ces valeurs. Donner le nombre de cycles d'horloge nécessaire pour obtenir ce PGCD avec votre circuit.

Ajouter à votre chronogramme, les signaux du registre d'état (les signaux q_i dans *iPC*) et des registres de données de la PO. Vérifier cycle par cycle le bon fonctionnement de votre circuit.

Ex. 4 : Implantation sur le FPGA

On va implanter le circuit PGCD sur la carte FPGA, en utilisant les interrupteurs pour entrer les valeurs des paramètres et les LEDS pour afficher le résultat.

Question 1 Donner le schéma correspondant au fichier `vhd/pgcdfpga.vhd`.

Sachant que la carte ne dispose que de 4 interrupteurs, 4 boutons poussoirs et 4 LEDs, expliquer l'intérêt de ce circuit. Retrouver notamment les boutons qui permettent de manipuler les premiers bits de B et celui qui opère l'affichage des bits de poids faibles ou forts.

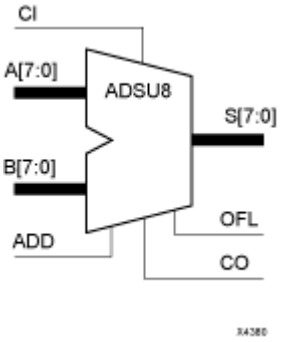
Question 2 Après avoir branché la carte correctement, implorer la puissance du FPGA par la commande :

```
$ make run_fpga TOP=PGCDFPGA
```

Vérifier sur carte le bon fonctionnement de votre circuit.

ADSU8

Macro: 8-Bit Cascadable Adder/Subtracter with Carry-In, Carry-Out, and Overflow



Introduction

When the ADD input is High, this element adds two 8-bit words (A7:A0 and B7:B0) and a carry-in (CI), producing, an 8-bit sum output (S7:S0) and carry-out (CO) or an overflow (OFL).
When the ADD input is Low, this element subtracts B7:B0 from A7:A0, producing an 8-bit difference output (S7:S0) and a carry-out (CO) or an overflow (OFL).
In add mode, CO and CI are active-High. In subtract mode, CO and CI are active-Low. OFL is active-High in add and subtract modes.

Logic Table

Input			Output
ADD	A	B	S
1	An	Bn	An+Bn+CI*
0	An	Bn	An-Bn-CI*
CI*: ADD = 0, CI, CO active LOW CI*: ADD = 1, CI, CO active HIGH			

Unsigned Binary Versus Two's Complement -This design element can operate on either 8-bit unsigned binary numbers or 8-bit two's-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as two's complement, the output can be interpreted as two's complement. The only functional difference between an unsigned binary operation and a two's-complement operation is the way they determine when "overflow" occurs. Unsigned binary uses CO, while two's complement uses OFL to determine when "overflow" occurs.
With adder/subtracters, either unsigned binary or two's-complement operations cause an overflow. If the result crosses the overflow boundary, an overflow is generated. Similarly, when the result crosses the carry-out boundary, a carry-out is generated.
Unsigned Binary Operation -For unsigned binary operation, this element can represent numbers between 0 and 255, inclusive. In add mode, CO is active

(High) when the sum exceeds the bounds of the adder/subtracter. In subtract mode, CO is an active-Low borrow-out and goes Low when the difference exceeds the bounds.
An unsigned binary "overflow" that is always active-High can be generated by gating the ADD signal and CO as follows:

```
unsigned overflow = CO XOR ADD
```

OFL is ignored in unsigned binary operation.
Two's-Complement Operation -For two's-complement operation, this element can represent numbers between -128 and +127, inclusive.
If an addition or subtraction operation result exceeds this range, the OFL output goes High. CO is ignored in two's-complement operation.

Design Entry Method

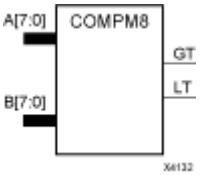
This design element is only for use in schematics.

For More Information

- See the *Spartan-3 Generation FPGA User Guide*.
- See the *Spartan-3 FPGA Family Data Sheet*.

COMPM8

Macro: 8-Bit Magnitude Comparator



Introduction

This design element is an 8-bit magnitude comparator that compare two positive Binary-weighted words. It compares A7 : A0 and B7 : B0, where A7 and B7 are the most significant bits.
The greater-than output (GT) is High when A > B, and the less-than output (LT) is High when A < B When the two words are equal, both GT and LT are Low. Equality can be measured with this macro by comparing both outputs with a NOR gate.

Logic Table

Inputs								Outputs	
A7, B7	A6, B6	A5, B5	A4, B4	A3, B3	A2, B2	A1, B1	A0, B0	GT	LT
A7>B7	X	X	X	X	X	X	X	1	0
A7<B7	X	X	X	X	X	X	X	0	1
A7=B7	A6>B6	X	X	X	X	X	X	1	0
A7=B7	A6<B6	X	X	X	X	X	X	0	1
A7=B7	A6=B6	A5>B5	X	X	X	X	X	1	0
A7=B7	A6=B6	A5<B5	X	X	X	X	X	0	1
A7=B7	A6=B6	A5=B5	A4>B4	X	X	X	X	1	0
A7=B7	A6=B6	A5=B5	A4<B4	X	X	X	X	0	1
A7=B7	A6=B6	A5=B5	A4=B4	A3>B3	X	X	X	1	0
A7=B7	A6=B6	A5=B5	A4=B4	A3<B3	X	X	X	0	1
A7=B7	A6=B6	A5=B5	A4=B4	A3=B3	A2>B2	X	X	1	0
A7=B7	A6=B6	A5=B5	A4=B4	A3=B3	A2<B2	X	X	0	1
A7=B7	A6=B6	A5=B5	A4=B4	A3=B3	A2=B2	A1>B1	X	1	0
A7=B7	A6=B6	A5=B5	A4=B4	A3=B3	A2=B2	A1<B1	X	0	1
A7=B7	A6=B6	A5=B5	A4=B4	A3=B3	A2=B2	A1=B1	A0>B0	1	0

Inputs								Outputs	
A7, B7	A6, B6	A5, B5	A4, B4	A3, B3	A2, B2	A1, B1	A0, B0	GT	LT
A7=B7	A6=B6	A5=B5	A4=B4	A3=B3	A2=B2	A1=B1	A0<B0	0	1
A7=B7	A6=B6	A5=B5	A4=B4	A3=B3	A2=B2	A1=B1	A0=B0	0	0

Design Entry Method

This design element is only for use in schematics.

For More Information

- See the *Spartan-3 Generation FPGA User Guide*.
- See the *Spartan-3 FPGA Family Data Sheet*.