TP2 : Initiation à l'environnement Modelsim

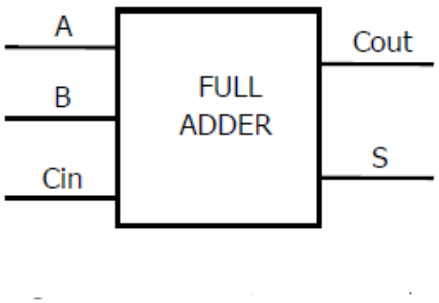
A. Objectifs

- Se familiariser avec Modelsim, le logiciel de simulation de Mentor Graphics
- L'utilisation du VHDL en logique combinatoire par description comportementale de l'architecture d'un Additionneur complet.
- Utiliser le VHDL en description structurelle pour concevoir un additionneur-soustracteur 4 bits avec détection du débordement.
- L'utilisation du VHDL pour décrire un banc de test (*Testbench*) de vérification.

B. Premier projet

1. Additionneur complet 1 bit

- A partir de la table de vérité de l'additionneur 1 bit on peut déterminer les équations simplifiées des sorties en fonction des entrées :



Inputs			Outputs	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

$$S = A \oplus B \oplus Cin$$
$$Cout = (A.B) + Cin.(A \oplus B)$$

2. Description VHDL du module et simulation (Modelsim)

- Créer dans un projet TP1 un fichier source VHDL **full_add.vhd** dans lequel il faut décrire l'additionneur complet. Le squelette du code VHDL de l'additionneur complet est fourni ci-après.
- Créer un chronogramme des signaux —*waveform*—; et séparer les signaux d'entrée de ceux de sortie.
- Lancer la simulation et interpréter le résultat.
- Analyser le code du fichier **full_add_tb.vhd** fourni sur ci-après, et déterminer les parties nécessaires pour créer des stimuli de test. Compléter les stimuli de test de façon à couvrir toutes les combinaisons possibles des entrées (A, B et Cin) répertoriées dans la table de vérité.
- Ajouter les deux fichiers (RTL et testbench) au projet TP1 et compiler.
  -- Description VHDL de l'additionneur complet

```
-- Bibliothèques
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--Entity
ENTITY full_add IS
-- ...
--Architecture
ARCHITECTURE behavior OF full_add IS
-- ...
```

-- TestBench de l'additionneur complet

```
-- Bibliothèques
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--Entity
ENTITY full_add_tb IS
END full_add_tb;
--Architecture
ARCHITECTURE test OF full_add_tb IS
-- Déclaration du composant à tester --
COMPONENT full_add
PORT(A : in std_logic;
B : in std_logic;
Cin : in std_logic;
S : out std_logic;
Cout : out std_logic);
END COMPONENT;
-- Déclaration des signaux de test --
signal A_tb, B_tb,Cin_tb, S_tb, Cout_tb: std_logic;
BEGIN
-- Instanciation du composant à tester -- DUT : Design Under Test
DUT: full_add PORT MAP (A=>A_tb,
B=>B_tb,
Cin=>Cin_tb,
S=>S_tb,
Cout=>Cout_tb);
-- Création des signaux de test --
process
BEGIN
A_tb <='0'; B_tb <='0'; Cin_tb <='0'; -- 000
wait for 20 ns;
A_tb <='1'; -- 100
wait for 20 ns;
A_tb <='0'; B_tb <='1'; -- 010
wait for 20 ns;
-- ...
END PROCESS;
END test;
```

C. Deuxième projet

1. Additionneur-soustracteur 4 bits

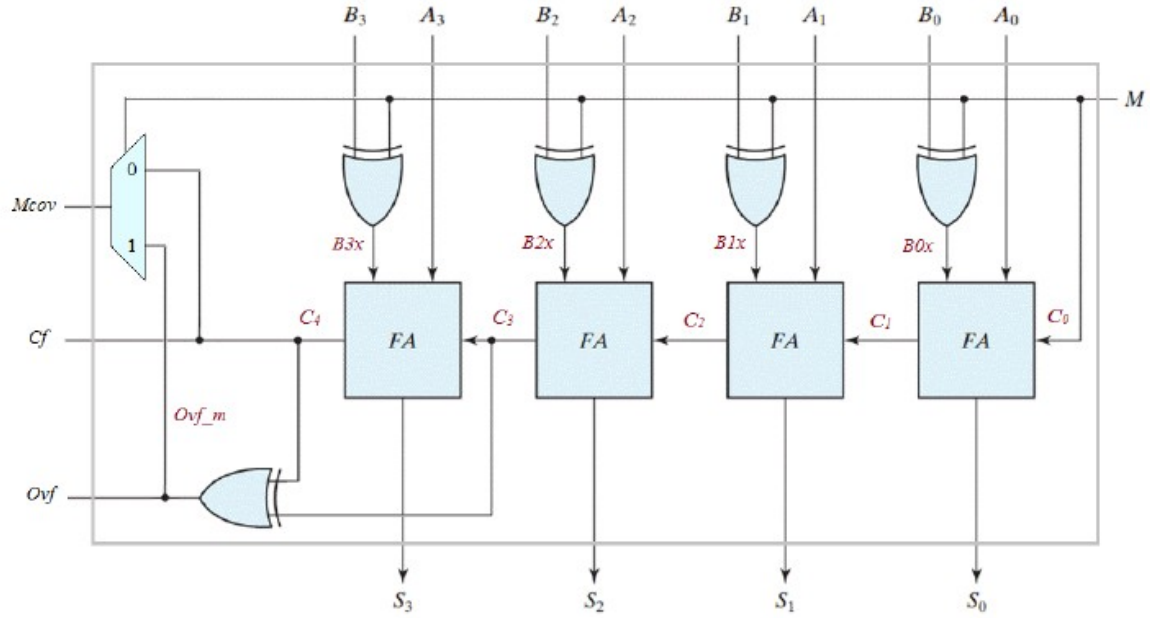
On désire décrire l'additionneur-soustracteur 4-bit représenté sur la figure. L'additionneur-soustracteur est un circuit capable d'additionner **ou** de soustraire deux nombres binaires, chacun de taille 4 bit, en fonction d'un signal de commande **M**.

Ce circuit permet de faire l'addition ou la soustraction de deux nombres A et B de 4 bits chacun. En

sortie on va avoir le résultat sur 4 bits ainsi que la retenu (*Cf*), le débordement (*Ovf*) et une sortie *Mcov*.

2. Structure de l'additionneur-soustracteur 4 bits

Le circuit de **l'additionneur-soustracteur 4-bit** à décrire est représenté par la figure suivante :



Mcov est la sortie d'un multiplexeur qui sélectionne **Cf** si l'opération réalisée est une addition ou **Ovf** si l'opération est une soustraction.

3. Description VHDL du module et simulation (Modelsim)

- Créer un fichier source VHDL **addsub4.vhd** pour décrire l'additionneur-soustracteur. Vous allez vous servir des modules suivants : un additionneur (**full_adder.vhd**), une porte XOR (**xor.vhd**) et un multiplexeur (**mux.vhd**) à l'intérieur de l'architecture de l'additionneur-soustracteur, faites attention aux noms des ports de **chaque** module déclaré (**component**) et instancié (Instance_X : nom_module **PORT MAP**(…,…,…)) afin de les connecter correctement aux signaux correspondants.
- Créer le testbench **addsub4_tb.vhd** et simulez avec ModelSim l'additionneur-soustracteur. Observez les chronogrammes et interprétez