République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Frères Mentouri – Constantine 1
Faculté des Sciences de la Technologie
Département d'Electronique

DOCUMENT DE TRAVAIL

Travaux Pratiques

MASTER 1, Instrumentation

VHDL - FPGA

Réalisé et présenté par :

Dr. Salah ABADLI

Année Universitaire : 2019 / 2020 (Semestre 2)

TP 1 : INITIATION A LA PROGRAMMATION VHDL

Première approche du logiciel simulateur de VHDL Altera Max+Plus II

Objectif : L'objectif primordial de ce premier TP est de se familiariser avec un outil de programmation des circuits logiques (numériques) programmables. Nous nous intéressons à un outil de la famille **Altera** : logiciel **Max+plus II**.

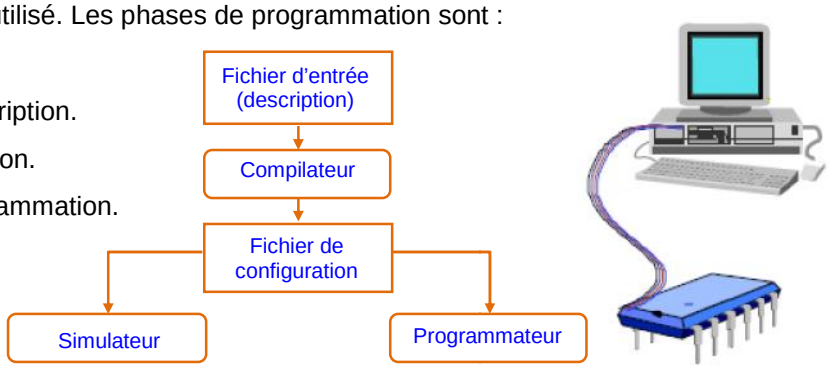
I- Synthèse : Circuits logiques programmables et outils de conception

Un circuit logique programmable se définit comme un composant électronique standard contenant des modules de logique combinatoires et séquentiels, dont les interconnexions internes sont désignées par programmation (interconnexions entre différentes cellules logiques ; voir cours). Il peut être configuré et reconfiguré par l'utilisateur, pour la réalisation de diverses fonctions logiques (numériques).

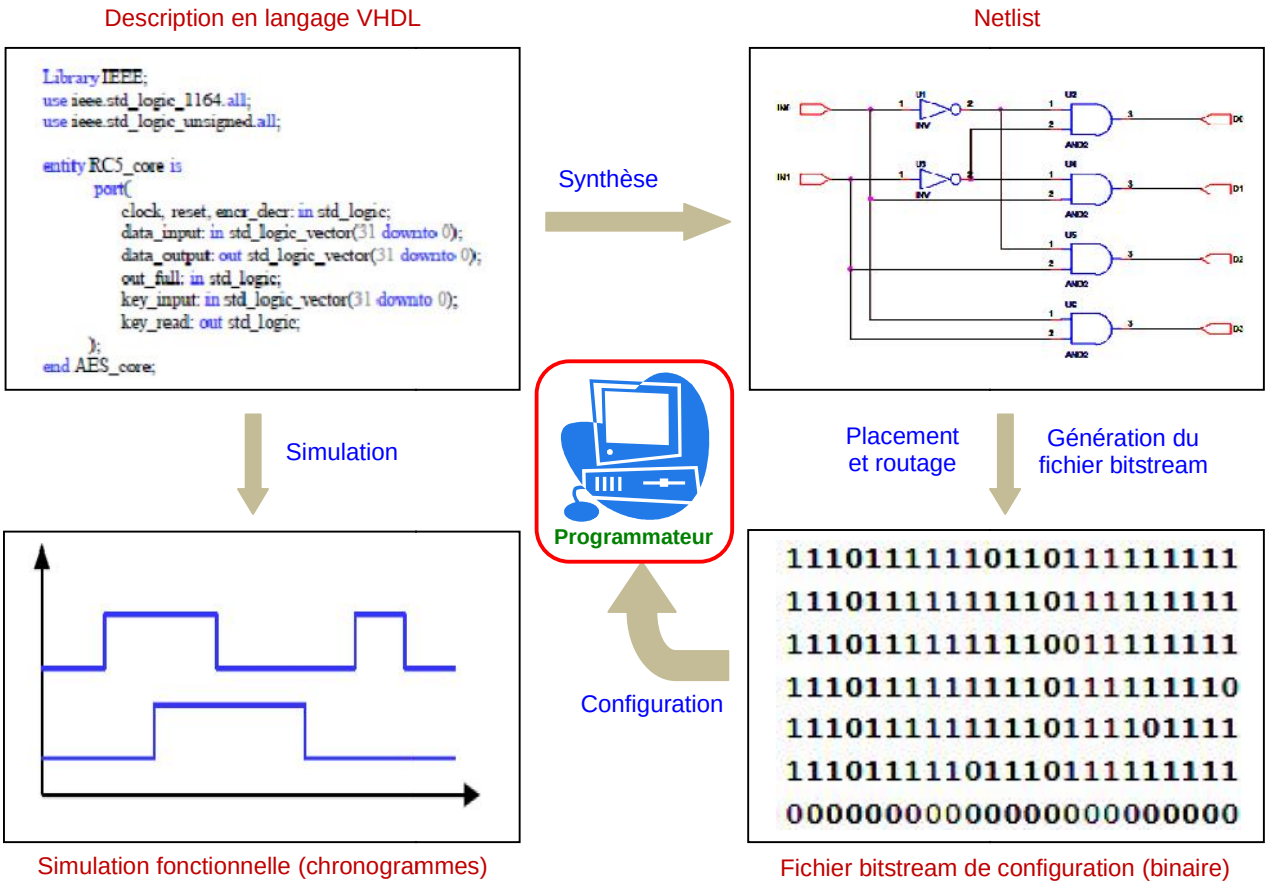
Plusieurs solutions sont possibles pour programmer un circuit logique programmable. Elles dépendent essentiellement de l'outil de développement (outil de conception) utilisé. Les phases de programmation sont :

- Saisie de la description du circuit logique.
- Compilation pour la vérification de la description.
- Simulation, synthèse (Netliste) et vérification.
- Insertion du circuit à programmer et programmation.

La figure ci-contre résume les différentes étapes de programmation d'un tel circuit.



Le synthétiseur de l'outil de conception assistée par ordinateur (CAO) génère dans un premier temps une **Netlist** qui décrit la connectivité de l'architecture. Puis l'outil de placement-routage place de façon optimale tous les composants et effectue le routage entre les différentes cellules logiques.



VHDL (**V**ery high-speed integrated circuits **H**ardware **D**escription **L**anguage) est un langage de description de matériel, destiné à représenter le comportement ainsi que l'architecture d'un système électronique logique ; en particulier des circuits numériques programmables (ASIC, FPGA, CPLD,…etc.).

Actuellement, le marché des circuits programmables est toujours en pleine croissance. Deux fournisseurs particuliers se disputent principalement ce marché : **Xilinx** et **Altera** (environ 80% du marché entre eux deux).

Par conséquence, les principaux fabricants de circuits logiques programmables proposent une version gratuite mais limitée de leurs outils de simulation et de synthèse. Pour débuter en VHDL, nous citons quelques environnements de programmation permettant la saisie d'un fichier VHDL.

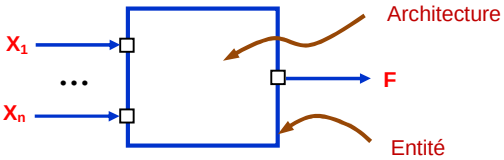
- Intel / Altera : **Max+plus II**, Quartus II, Modelsm,…etc.
- Xilinx : ISE,...etc.

II- Structure générale d'une description VHDL

D'une manière générale, un système numérique est vu comme une "boîte noire". En VHDL, la "boîte noire" est nommée **entité** (entity) et une entité doit toujours être associée avec au moins une description de son contenu, de son implémentation : c'est l'**architecture**. Une description VHDL est donc composée de deux parties indissociables :

- L'**entité** (entity) : elle décrit l'extérieur l'interface du composant et permet ainsi de définir les entrées/sorties. Pour déclarer l'entité on donne un nom et on précise la liste des différents signaux d'entrés/sorties (ports d'E/S).
- L'**architecture** (architecture) : décrit l'intérieur du composant. Elle contient les instructions VHDL permettant de réaliser le fonctionnement attendu. Elle comporte une partie de déclaration et un corps de programme.

La structure générale d'un programme en VHDL est la suivante :



déclaration des bibliothèques IEEE

```
library ieee;
use ieee.std_logic_1164.all;
.......
```

```
-- Pour écrire des commentaires
```

déclaration des entrées/sorties

```
entity NOM_ENTITE is
 port(
       .......
    );
end NOM_ENTITE ;
```

Nom de l'entité

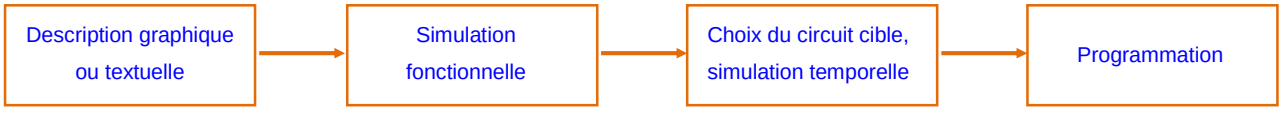
déclaration de l'architecture

```
architecture NOM_ARCHITECTURE of NOM_ENTITE is
 begin
       .......
 end NOM_ARCHITECTURE ;
```

Nom de l'architecture

III- Présentation du logiciel

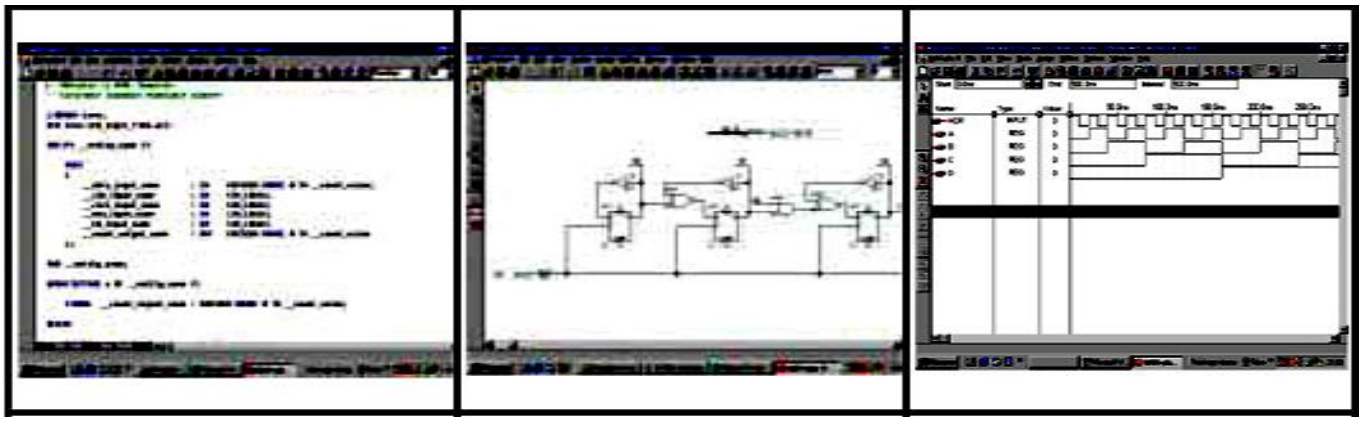
Le logiciel **Max+plus II** permet entre autres, la description d'un projet (système numérique), sa compilation, sa simulation logique et temporelle, son analyse temporelle et la programmation d'un circuit cible (CPLD ou FPGA).



La description du système numérique (logique) peut être faite à l'aide d'une des entrées suivantes :

- Editeur de texte : pour l'utilisation du langage VHDL ou VERILOG.
- Editeur graphique : permet d'utiliser les composants prédéfinis des bibliothèques fournies par le logiciel.
- Editeur de chronogrammes : avec lequel on représentera l'évolution temporelle et celle attendue des sorties.

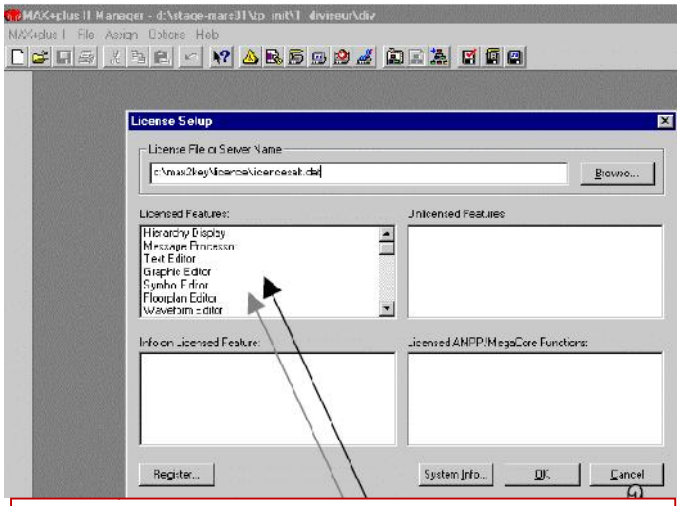
Editeur de texte | Editeur graphique | Editeur de chronogrammes



A chaque description est associé un symbole graphique du composant ainsi réalisé. L'éditeur graphique permettra alors de relier éventuellement ces composants les uns aux autres. Chaque sous-ensemble puis le système global est ensuite compilé, puis simulé par le simulateur logique, puis analysé et envoyé vers le circuit cible via le programmateur.

III-1- Installation du logiciel Max+plus II

Pour installer Max+plus II, télécharger depuis le serveur Altéra et suivre les instructions pour une installation complète (Full installation). Installer le fichier licence. Ce dernier est un fichier texte récupéré par e-mail, auprès du service Altera moyennant la fourniture d'un numéro d'identification de disc dure. Il faut donc changer l'extension du fichier (**license.dat**). Copier ce fichier dans le **C:** de votre PC et déclarer ensuite le chemin dans le logiciel (**Option/License setup/Browse**).



Toutes les possibilités du logiciel sont utilisables

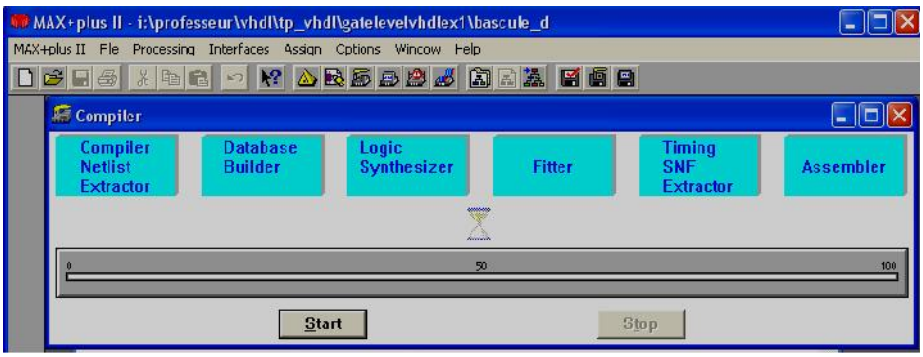
L'environnement de démarrage du logiciel Max+plus II est le suivant :

La création d'une description VHDL sous Max+plus II nécessite la génération, par le logiciel, d'un nombre important de fichiers. Tous les fichiers relatifs à une description portent le même nom, mais une extension différente.

III-2- Début d'un projet

Pour débuter un projet, il faut suivre les 4 étapes suivantes :

- **Etape de conception** : consiste à la création et l'activation d'un nouveau projet. Il faut sélectionner d'abord un nom de projet et une directory ou seront stockés les fichiers (pour nos TP, **C:\Max2work\Nom de projet**).
  - ✓ Créer un nouveau fichier ; sous l'éditeur de texte (**Text Editor file**).
  - ✓ Sauvegarder le fichier (**Save as**) ; avec le même nom de l'entité du programme VHDL et l'extension "**.vhd**".
  - ✓ Activer le projet (**Set projet to current file**). En effet, le logiciel ne traite qu'un projet à la fois ; il faudra vérifier que le projet en cours de traitement est bien celui désiré.
- **Etape de saisie du code VHDL** : une fois la première étape terminée, saisissez le code de description VHDL.
  - ✓ Déclaration des bibliothèques.
  - ✓ Déclaration de l'entité.
  - ✓ Déclaration de l'architecture.
- **Etape de compilation** : une fois la description faite, il faut vérifier que cette description est correcte et pour cela on utilise la phase de compilation (**Max+plus II → Compiler**). Vous avez à ce moment, accès à cette fenêtre :

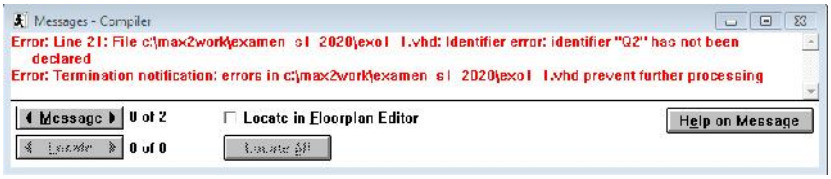


  - ✓ S'il y a des erreurs il faut les corrigées.
  - ✓ Lorsque la compilation passe sans erreurs, il est alors possible de vérifier le comportement logique du circuit.

Remarque :

Pendant l'étape de compilation, on distingue trois types de couleurs des messages affichées :

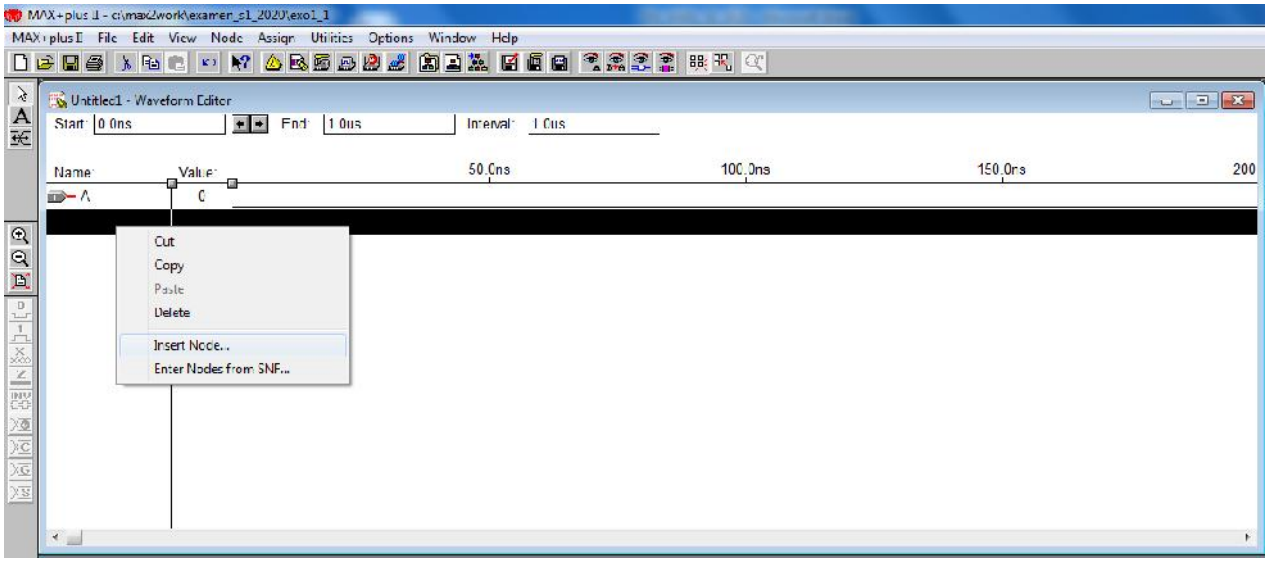
- **Messages de couleur rouge** : erreurs obligatoires à corriger (erreurs de syntaxe ou de synthèse). Dans ce cas, le code VHDL ne peut pas être compilé et ne peut pas être synthétisé (code VHDL non exécutable).



- **Messages de couleur bleu** : erreurs non obligatoires à corriger (erreurs d'avertissement ou des "warnings"). Dans ce cas, le code VHDL peut être compilé et peut être synthétisé (code VHDL exécutable).

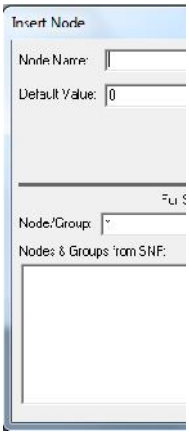
- **Messages de couleur verte uniquement** : pas d'erreurs. Le code VHDL est parfaitement exécutable.

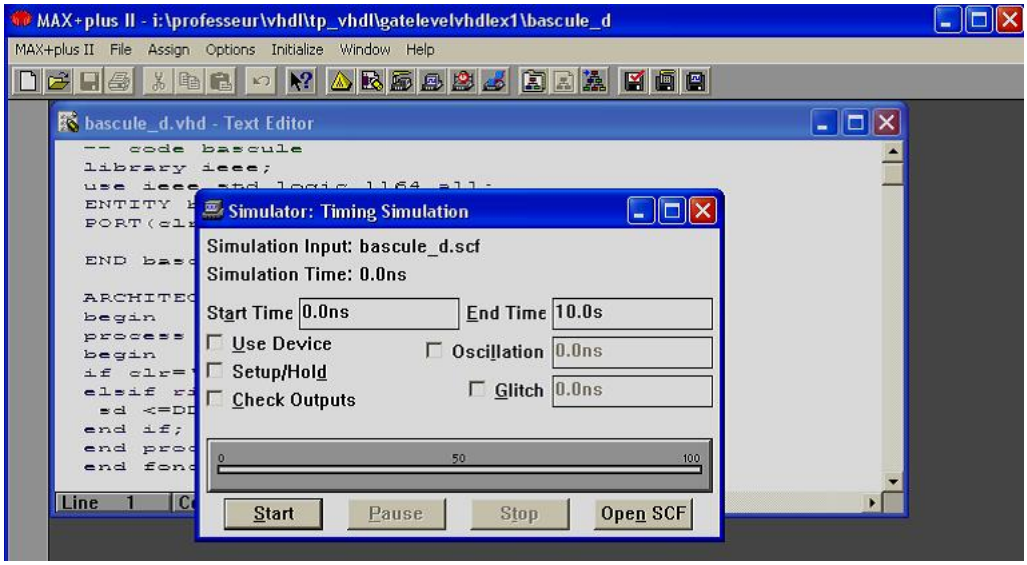
- **Etape de vérification** : comporte la simulation de type **Waveform Editor**, pour visualiser les chronogrammes.



Sur le fichier de type Waveform, nous affectons des valeurs pour les entrées de l'entité et on examine les résultats des sorties.

- ✓ Sauvegarder le fichier sous le même nom de l'entité avec l'extension "**.scf**".
- ✓ Sur la fenêtre affichée, avec la souris, faite entrer les nœuds (**Insert Node**) pour ajouter les signaux voulus.
- ✓ Configurer le temps de simulation des signaux (**File → End Time**).
- ✓ Configurer le pas de visualisation des signaux (**Options → Grid Size**).
- ✓ Entrer des valeurs de votre choix pour les entrées déclarés.
- ✓ Cliquer sur l'icône **Simuler** pour la simulation, puis cliquer sur **Start** et observer les chronogrammes (à l'aide du bouton **OpenSNF**) et vérifier vos résultats.

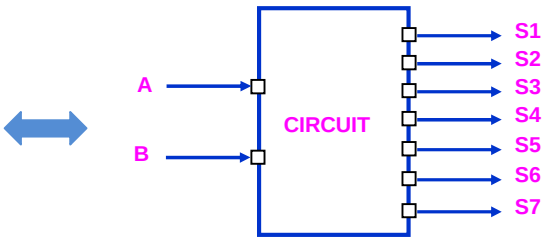




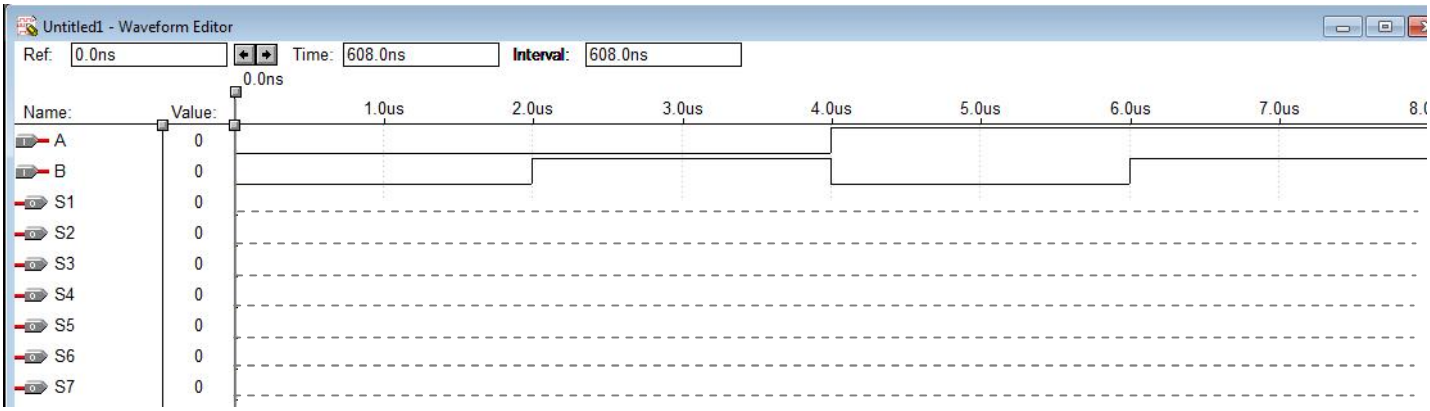
IV- Exercice d'application

Soit le code de description VHDL du circuit numérique suivant :

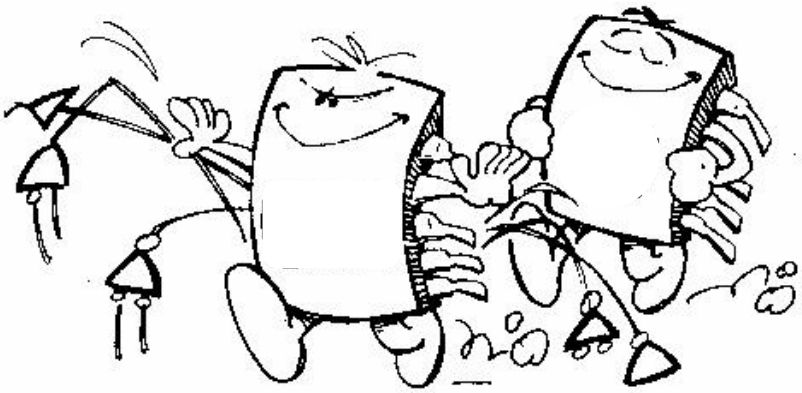
```
library ieee;
use ieee.std_logic_1164.all;

entity CIRCUIT is
    port (A,B :in std_logic;
          S1,S2,S3,S4,S5,S6,S7:out std_logic);
end CIRCUIT;

architecture DESCRIPTION of CIRCUIT is
    begin
      S1 <= A and B;
      S2 <= A or B;
      S3 <= A xor B;
      S4 <= not A;
      S5 <= A nand B;
      S6 <= A nor B;
      S7 <= not(A xor B);
end DESCRIPTION;
```



- Faire les étapes mentionnées au paragraphe III-2 précédent : Saisir le code VHDL du circuit donné, sauvegarder, compiler, corriger les erreurs si nécessaire, simuler et vérifier les résultats via waveform editor ?
- Tracer les chronogrammes des sorties pour le cas ci-dessous (simulation de 0 à 8 us avec un pas de 1 us) ?



- Répondez aux questions dans le rapport d'évaluation donné à la salle de TP.



TP2 : Exploitation du simulateur de VHDL

Additionneur Complet, Multiplexeur et Démultiplexeur

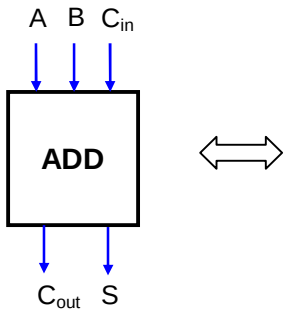
Noms : …… …… **Note :** ……

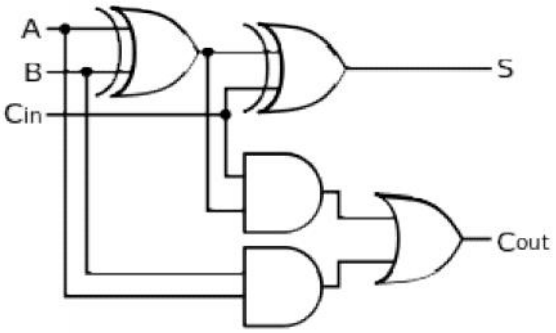
Remarque : **N'oublier pas de rendre le rapport de TP en fin d'exercice.**

Objectif : L'objectif essentiel de ce 2ème TP est toujours de bien maitriser la programmation VHDL via l'outil Altera Max+plus II. Nous nous intéressons à l'exploitation du simulateur VHDL pour la vérification des fonctions logiques.

Exercice 1 : Additionneur complet de 2 mots de 1 bit (affectation inconditionnelle)

Ecrire le code de description VHDL qui permet de réaliser un additionneur complet 1 bit, comme le montre la figure suivante :

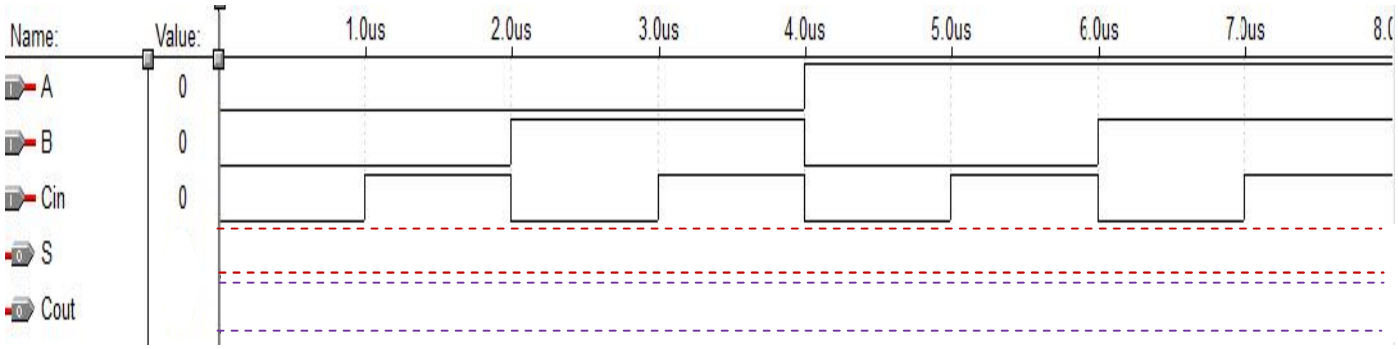




A	B	C_{in}	S	C_{out}
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

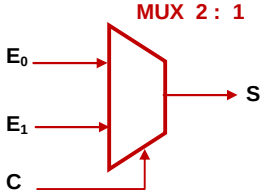
1- Sauvegarder, compiler, corriger les erreurs si nécessaire, simuler et vérifier les résultats via Waveform editor ?

2- Visualiser les chronogrammes des entrées/sorties (temps de 0 à 8 µs avec un pas 1 µs) et remplir le tableau ?



Exercice 2 : Multiplexeur 2 vers 1 (affectation inconditionnelle)

Ecrire le code de description VHDL qui permet de réaliser le multiplexeur 2 vers 1 de la figure suivante :

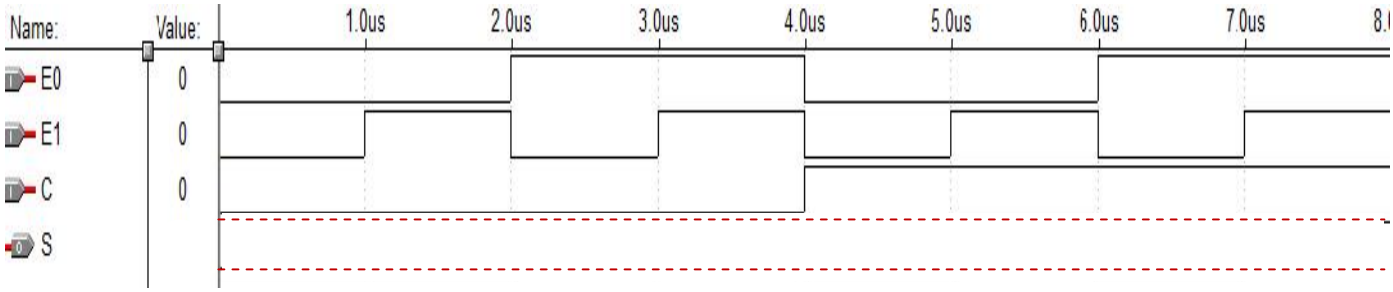


C	S
0	E_0
1	E_1

$$S = \overline{C}\, E_0 + C\, E_1$$

1- Sauvegarder, compiler, corriger les erreurs si nécessaire, simuler et vérifier les résultats via Waveform editor ?

2- Visualiser les chronogrammes des entrées / sorties (temps de 0 à 8 µs avec un pas 1 µs) et remplir le tableau ?



C / E_0 E_1	0 0	0 1	1 0	1 1
0				
1				

3- Donner le code de description VHDL de ce circuit ?

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

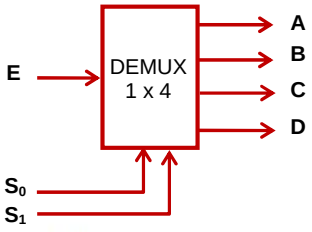
. . . . . .

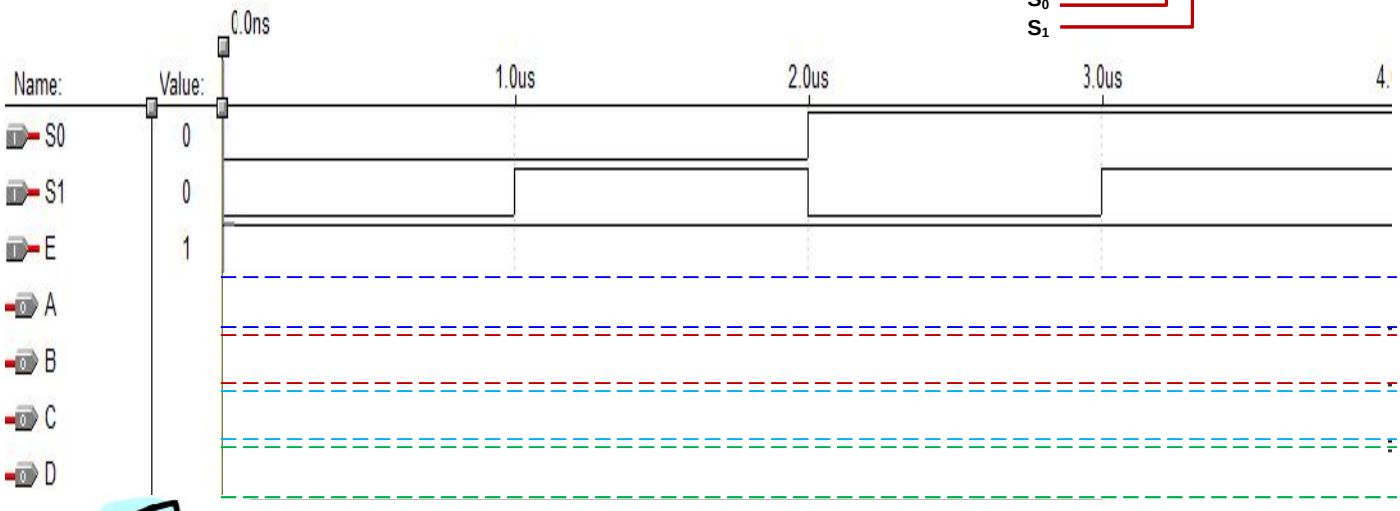
Exercice 3 : Démultiplexeur 1 vers 4 (affectation conditionnelle)

1- Ecrire et compiler le code de description VHDL du circuit de la figure en utilisant l'instruction When ... else ... ?

2- Simuler le code VHDL réalisé pour le cas suivant :

E = 1 pour les 4 situations suivantes : S_0S_1 = 00, 01, 10 et 11 (voir figure).





Réalisé par : Dr. S. ABADLI.

TP3 : Développement de quelques exemples de circuits en VHDL

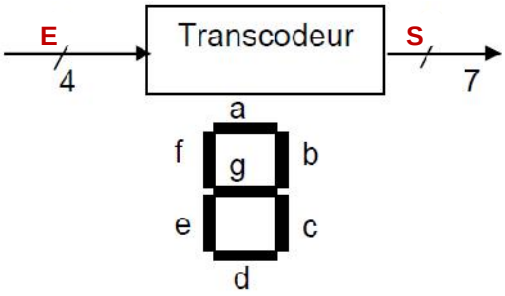
Compteur décimal et Multiplexeur 4 vers 1

Remarque : **N'oublier pas de rendre le rapport de TP en fin d'exercice dans la salle de TP.**

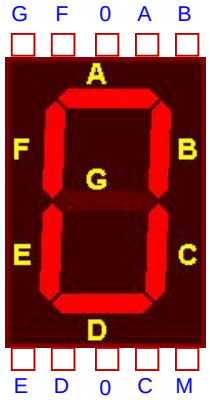
Objectif : L'objectif essentiel de ce 3ème TP est toujours l'écriture de programmes de description VHDL, l'exécution de ces programmes, la simulation fonctionnelle et l'interprétation des résultats. Nous nous intéressons dans ce TP, en particulier, aux instructions concurrentes en VHDL (la gestion de l'afficheur 7 segments, le multiplexeur,…etc.).

Exercice 1 : instruction concurrente WITH-SELECT (affectation sélective ; ex. Compteur décimal)

A l'aide de l'instruction d'appel WITH . . . SELECT . . . WHEN . . ., écrire le code de description VHDL qui permet de réaliser le transcodeur (convertisseur) 4 vers 7 segments de la figure suivante :



e3	e2	e1	e0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

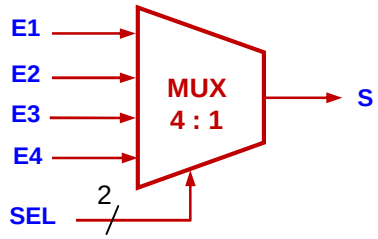


1- Sauvegarder, compiler et corriger les erreurs si nécessaire. Simuler et vérifier les résultats via Waveform editor.

2- Visualiser les chronogrammes des entrées/sorties en binaire (temps de 0 à 10 µs avec un pas 1 µs) ?

Exercice 2 : instruction concurrente WHEN-ELSE (affectation conditionnelle ; ex. Multiplexeur 4 : 1)

En utilisant l'instruction concurrente WHEN ... ELSE ..., écrire le code de description VHDL qui permet de réaliser un multiplexeur 4 vers 1 (MUX 4 : 1) ; comme le montre la figure ci-contre.



1- Saisir le code, sauvegarder, compiler et corriger les erreurs si nécessaire.

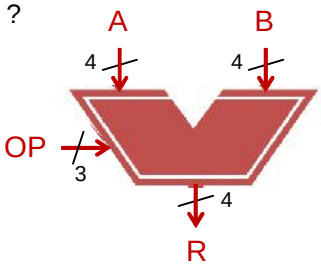
2- Simuler et vérifier les résultats des entrées/sorties via Waveform editor.

Rappel : SEL = 00 : S ← E1 SEL = 01 : S ← E2 SEL = 10 : S ← E3
SEL = 11 : S ← E4

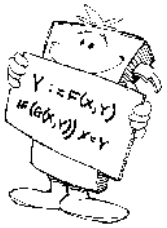
3- Visualiser les chronogrammes des E/S (temps de 0 à 16 µs avec un pas 1 µs) ?

Exercice 3 : instruction CASE (affectation sélective ; ex. Unité Arithmétique et Logique UAL 4-bits)

Donner le code de description VHDL de l'Unité Arithmétique et Logique (UAL 4-bits), de la figure suivante, en utilisant l'instruction d'assignation CASE ?



OP	Fonction réalisée
000	A + B
001	A - B
010	A + 1
011	A - 1
100	A AND B
101	A OR B
110	NON A
111	A XOR B



1- Saisir le code, sauvegarder, compiler et corriger les erreurs si nécessaire.

2- Simuler et vérifier les résultats des entrées/sorties via Waveform editor.

3- Refaire l'exercice en utilisant l'instruction d'assignation With . . . Select . . . When . . . ?

Réalisé par : Dr. S. ABADLI.

TP4 : Développement de quelques exemples de circuits numériques en VHDL

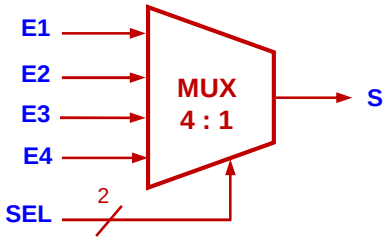
MULTIPLEXEUR ET COMPTEURS BINAIRES

Remarque : **N'oublier pas de rendre le rapport de TP en fin d'exercice dans la salle de TP.**

Objectif : L'objectif primordial de ce 4ème TP est toujours l'exécution de programmes de description VHDL et l'interprétation des résultats ; via l'outil Altera Max+plus II. Nous nous intéressons maintenant à l'utilisation de processus (process) en VHDL et aux instructions séquentielles internes aux processus et aux procédures.

Exercice 1 : Structure IF . . . THEN . . . ELSE . . . END IF (exemple Multiplexeur 4 vers 1)

En utilisant un process, avec liste de sensibilité, écrire le code de description VHDL qui permet de construire un multiplexeur 4 vers 1 (MUX 4 : 1) ; comme le montre la figure ci-contre.



1- Saisir le code, Sauvegarder, Compiler, Corriger les erreurs si nécessaire.

2- Simuler et Vérifier les résultats via Waveform editor (temps entre 0 et 16 µs).

3- Visualiser les chronogrammes des entrées / sorties et discuter les résultats.

Exercice 2 : Unité cadencée par une horloge H (ex. Compteur binaire 4 bits simple)

A) Ecrire le code de description VHDL qui permet de réaliser un compteur binaire 4 bits, comme le montre la figure. Le cahier des charges est le suivant :

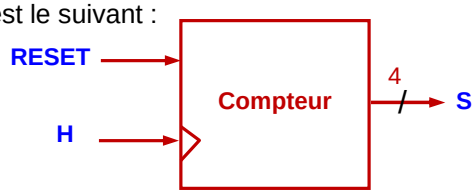
- Au front d'horloge "H", le circuit réalise une incrémentation de 1.
- Entre 2 fronts d'horloge, la sortie S est figée ou bien conservée.

1- Sauvegarder, compiler, corriger les erreurs si nécessaire, simuler et vérifier les résultats via Waveform editor ?

2- Visualiser les chronogrammes de l'entrée et la sortie (temps de 0 à 32 µs avec un pas 1 µs) ?

B) Modifier le code de description VHDL précédent pour permettre de réaliser un compteur binaire 4 bits avec RESET synchrone ; comme le montre la figure. Le cahier des charges est le suivant :

- Si Reset = 1 alors S = 0 (remise à zéro).
- Au front d'horloge "H", le circuit réalise une incrémentation de 1.
- Entre 2 fronts d'horloge, la sortie S est figée ou bien conservée.



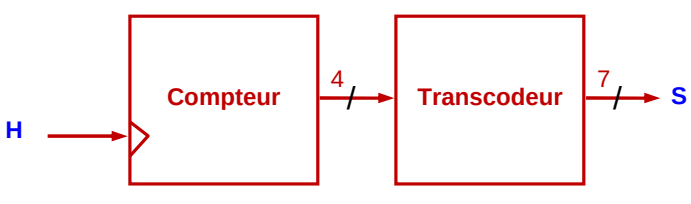
1- Sauvegarder, Compiler, Corriger les erreurs si nécessaire, Simuler et Vérifier les résultats via Waveform editor ?

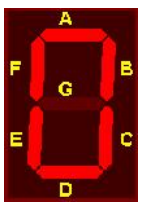
2- Visualiser les chronogrammes de l'entrée et la sortie (temps de 0 à 32 µs avec un pas 1 µs) ?

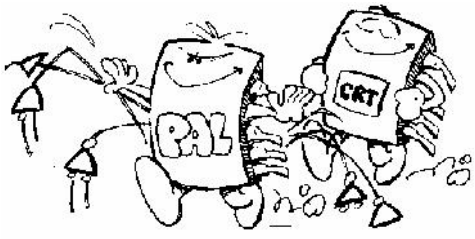
Exercice 3 : Système cadencée par une horloge H (exemple Compteur binaire 4 bits & afficheur 7 seg.)

Ecrire le code de description VHDL qui permet de réaliser un compteur binaire 4 bits attaché à un transcodeur 4 vers 7 segments ; comme le montre la figure ci-dessous. Le cahier des charges est le suivant :

- Au front d'horloge "H", le circuit réalise une incrémentation de 1.
- Entre 2 fronts d'horloge, la sortie S est figée ou bien conservée.







Réalisé par : Dr. S. ABADLI.

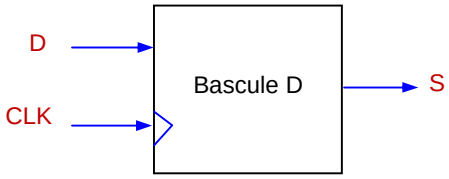
TP5 : Développement de quelques exemples de circuits numériques en VHDL

BASCULES, REGISTRES A DECALAGE ET SYSTEMES SEQUENTIELS

Remarque : **N'oublier pas de rendre le rapport de TP en fin d'exercice dans la salle de TP.**

Objectif : L'objectif primordial de ce 5ème TP est toujours l'exécution de programmes de description VHDL et l'interprétation des résultats, via l'outil Altera Max+plus II. Nous nous intéressons beaucoup plus dans ce TP aux instructions séquentielles qui se déclarent dans le corps d'un process et aux systèmes séquentiels en VHDL.

Exercice 1 : Bascule D simple



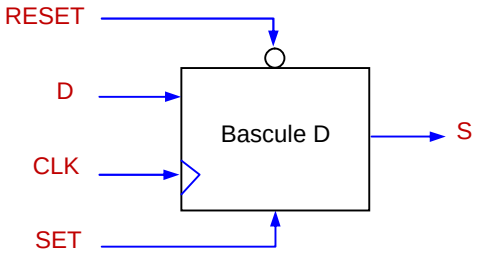
Ecrire le code de description VHDL qui permet de réaliser une bascule D.

1- Saisir le code, Sauvegarder, Compiler, Corriger les erreurs si nécessaire.

2- Simuler et Vérifier les résultats via Waveform editor ?

3- Exemple, visualiser les chronogrammes des E/S pour un temps de 0 à 6 µs avec un pas 1 µs et ≠ possibilités ?

Exercice 2 : Bascule D avec SET et RESET



Ecrire le code de description VHDL qui permet de réaliser une bascule D avec SET / RESET.

Si RESET = 1 alors S = 0,

Si SET = 1 alors S = 1, Sinon S = D sur front montant de CLK

1- Saisir le code, Sauvegarder, Compiler, Corriger les erreurs si nécessaire.

2- Simuler et Vérifier les résultats via Waveform editor ?

3- Visualiser les chronogrammes des entrées / sorties ?

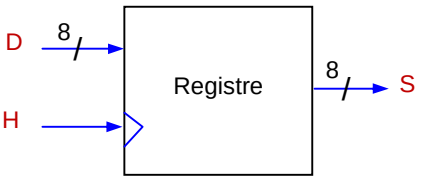
4- Est-il possible d'écrire le code VHDL de ce circuit sans process ?

5- Quelle est la liste des sensibilités dans le cas de process ?

6- Le RESET est synchrone ou asynchrone ?

7- Visualiser et discuter les chronogrammes des entrées / sorties ; pour diverses possibilités : D, SET, RESET ?

Exercice 3 : Registre parallèle 8 bits



Ecrire le code de description VHDL qui permet de réaliser un registre parallèle.

Le cahier des charges est le suivant :

- Au front d'horloge, le bus de sortie Q recopie le bus d'entrée D.
- Entre 2 fronts d'horloge, le bus de sortie Q est figé (conservé ou mémorisé).

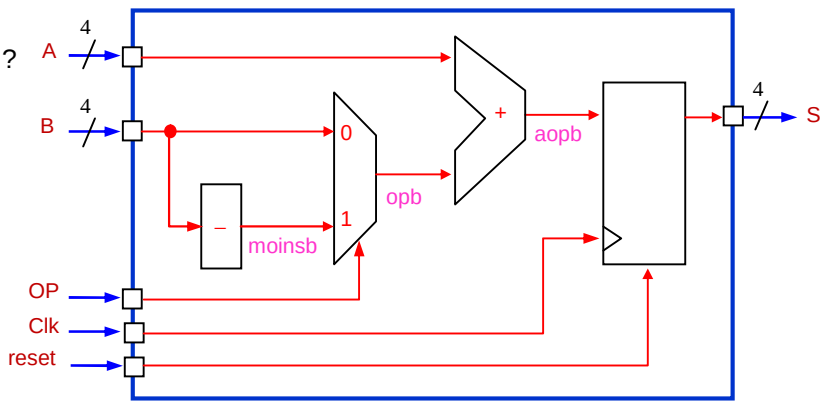
Exercice 4 : Processus et signaux (Additionneur/Soustracteur de 2 mots de 4 bits)

Soit le circuit numérique de la figure suivante :

1- Ecrire et compiler le code VHDL de ce circuit ?

2- Visualiser les chronogrammes des E/S ?

3- Vérifier les résultats via la simulation ?



Réalisé par : Dr. S. ABADLI.

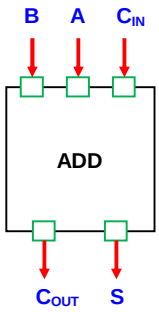
TP6 : Quelques exemples de circuits numériques en VHDL
COMPOSANTS, SIGNAUX D'INTERCONNEXIONS ET INSTANCIATIONS

Remarque : **N'oublier pas de rendre le rapport de TP en fin d'exercice dans la salle de TP.**

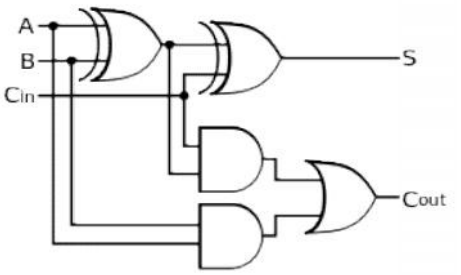
Objectif : L'objectif primordial de ce 6ème TP est toujours l'exécution de programmes de description VHDL et l'interprétation des résultats, via l'outil Altera Max+plus II. Particulièrement, nous nous intéressons dans ce TP aux composants, aux signaux d'interconnexions des composants et aux instanciations.

Exercice 1 : Additionneur complet de deux mots binaires (à 1 bit et à 2 bits)

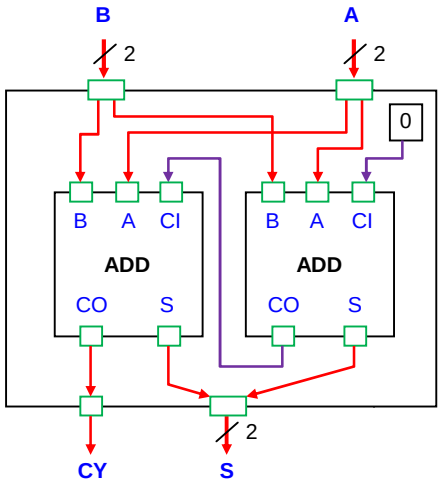
1- Ecrire le code de description VHDL d'un circuit additionneur complet (ADD) de 2 mots à 1 bit.
Visualiser et discuter les résultats des E/S ?



2- En utilisant le circuit ADD comme composant, écrire le code de description VHDL d'un circuit additionneur complet de 2 mots à 2 bit.
Visualiser et discuter les résultats des E/S ?

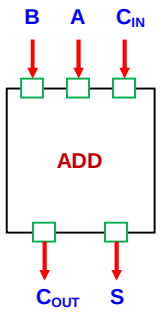


Circuit logique **ADD**

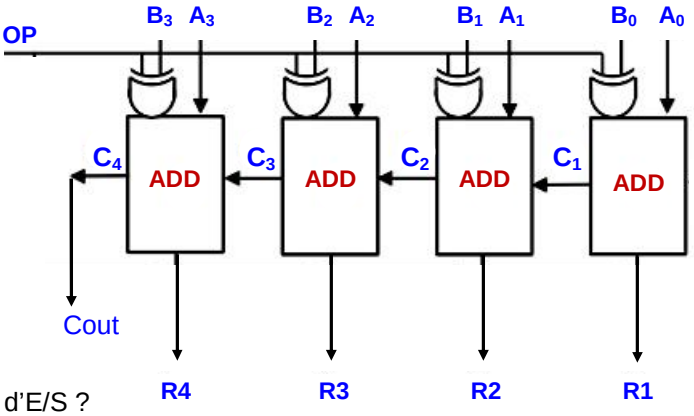


Exercice 2 : Additionneur / Soustracteur complet de deux mots binaires à 4 bits

En utilisant le circuit ADD comme composant, écrire le code de description VHDL du circuit suivant :



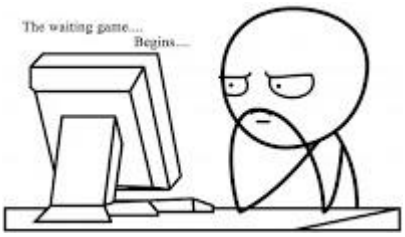
Op = 0 → Additionneur
Op = 1 → Soustracteur

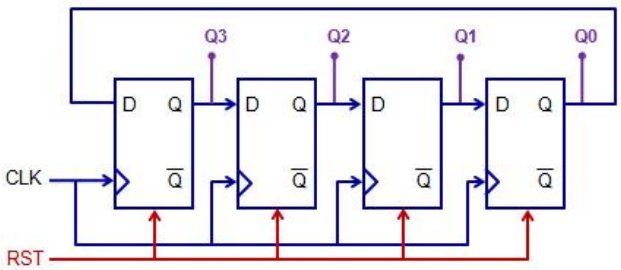


Vérifier et discuter les résultats via les chronogrammes d'E/S ?

Exercice 3 : Circuit logique à base de 4 bascules D

Ecrire le code de description VHDL du circuit suivant :
Quelle est la fonction de ce circuit logique ?





Réalisé par : Dr. S. ABADLI.