AutoSIN

—

TP2 : Logique combinatoire en VHDL

2025 – 2026

Objectifs de la séance

- Appliquer les différentes étapes du flot de conception numérique,
- Vérifier en simulation que l'étape de conception est correcte,
- Configurer un FPGA et vérifier que le design implanté fonctionne correctement.

Matériel fourni

Pour les exercices 2 à 6, vous réaliserez les étapes de conception et de vérification en utilisant les feuilles à disposition.

Pour la mise en œuvre finale, vous disposez de cartes de développement Altera DE1 et du câble USB associé permettant de les connecter au PC pour les configurer.

Validation

Lorsque vous avez terminé un exercice, appelez l'enseignant afin qu'il valide avec vous le bon fonctionnement du circuit.

Flot de conception numérique sur FPGA

Les étapes de la méthodologie de conception numérique sur FPGA sont représentées en Figure 1.

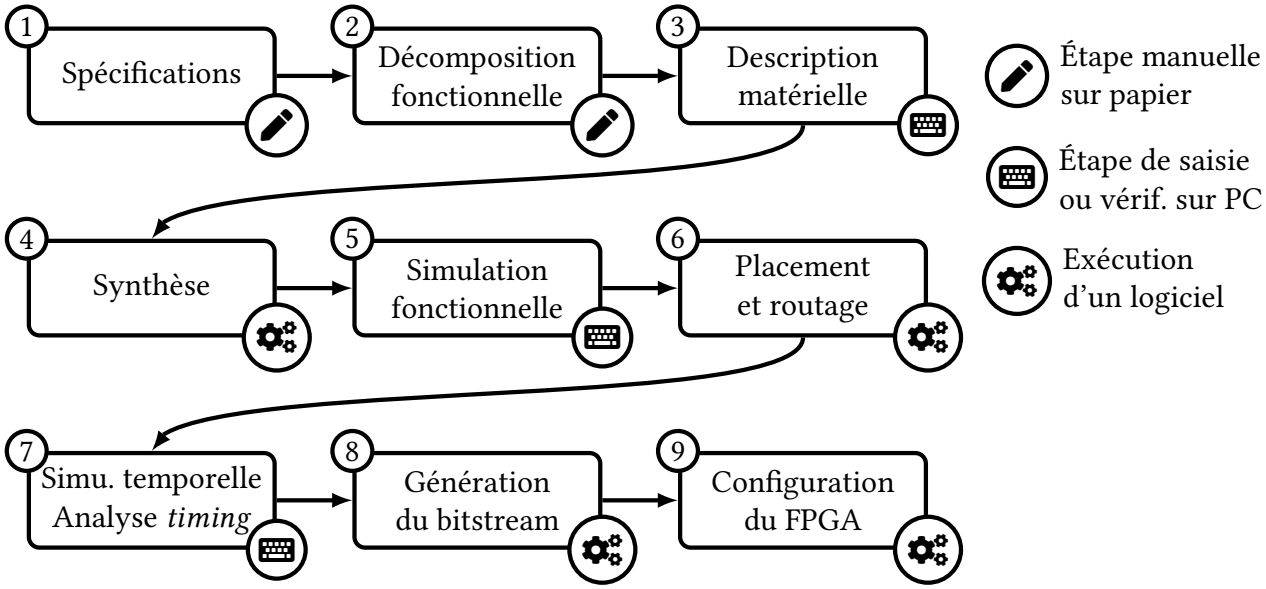


Figure 1 – Flot de conception numérique sur FPGA

① L'étape de **spécifications** consiste à décrire **en français** ce que doit faire le système,

② L'étape de **décomposition fonctionnelle** consiste à décomposer si nécessaire puis à représenter le design complexe en blocs plus simples. Chaque bloc sera nommé, et les entrées et sorties également,

③ L'étape de **description matérielle** consiste à décrire les blocs identifiés à l'étape précédente, puis à les connecter entre eux. Ceci est fait en utilisant un langage de description matérielle, comme le VHDL, ou en saisissant directement le schéma dans un logiciel dédié,

④ L'étape de **synthèse** consiste, à faire analyser la description par un logiciel dédié, qui sélectionnera automatiquement les composants logiques nécessaires à la mise en œuvre du design,

⑤ L'étape de **simulation fonctionnelle** consiste à vérifier que l'étape de synthèse a bien produit un design conforme aux spécifications d'un point de vue de la **fonctionnalité**, c'est à dire sans tenir compte des délais de propagation dans les éléments logiques,

⑥ L'étape de **placement et routage** consiste à placer les composants logiques préalablement sélectionnés dans un plan 2D et à réaliser les interconnexions nécessaires. Cette étape donne le coût du design en ressources logiques : les LUTs et interconnexions.

⑦ L'étape de **simulation temporelle** consiste à vérifier que l'étape de placement et routage a bien produit un design conforme aux spécifications, mais cette fois en tenant compte des délais de propagation dans les éléments logiques. Cela permet de déduire la fréquence maximale de fonctionnement du circuit grâce à l'analyse de *timing*.

⑧ L'étape de **génération du bitstream** consiste générer le fichier binaire de configuration du FPGA, qui contient notamment les valeurs à stocker dans les LUTs,

⑨ L'étape de **configuration** consiste finalement à configurer le FPGA avec le bitstream préalablement généré et à vérifier en pratique que le design fonctionne comme attendu.

Exercice 1 : Comparaison de deux entrées

Afin de s'approprier le flot de conception numérique décrit précédemment, nous allons l'appliquer à un exemple volontairement simple.

1.1 Spécifications

Concevoir un circuit à deux entrées, `a` et `b`, dont la sortie `s` prend la valeur `1` si et seulement si les deux entrées sont différentes.

1.2 Décomposition fonctionnelle

Vu la simplicité du design, on n'utilisera qu'un seul bloc fonctionnel, représenté en Figure 2.

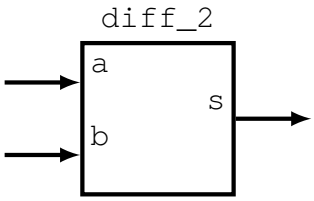


FIGURE 2 – Décomposition fonctionnelle du circuit à réaliser

1.3 Description matérielle

La description matérielle en VHDL de la fonction XOR à deux entrées est rappelée en Figure 3a. Quelques règles à respecter pour écrire du code VHDL lisible sont données en Figure 3b.

diff_2.vhd

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY diff_2 IS

  PORT (
    a : IN  STD_LOGIC;
    b : IN  STD_LOGIC;
    s : OUT STD_LOGIC);

END diff_2;

ARCHITECTURE archi OF diff_2 IS

BEGIN

  s <= a XOR b;

END archi;
```

(a) Fonction XOR à deux entrées

- Nom identique pour le projet, le fichier et l'entité,
  - Projet : `diff_2`
  - Fichier : `diff_2.vhd`
  - Entité : **`ENTITY diff_2 IS`**
- Mots-clés du langage en MAJUSCULE,
  - `LIBRARY`, `USE`, `ENTITY`
- Identifiants bien choisis, en minuscule, et ne commençant pas par un chiffre,
  - `s`, `diff_2`
- Indentation cohérente,
- Alignement des entrées / sorties,
- Alignement des affectations,

(b) Règles de codage à suivre en VHDL

FIGURE 3 – Fonction XOR et règles de codage en VHDL

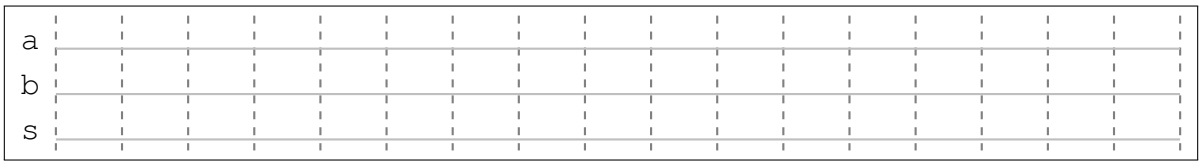
Afin de réaliser la description matérielle, lancer le logiciel Quartus 9.1. Créer ensuite un nouveau projet puis créer un nouveau fichier source VHDL. Décrire ensuite le circuit en VHDL.

1.4 Synthèse

Synthétiser le design, c'est à dire le faire analyser par le logiciel de conception, en cliquant sur *Analysis & Synthesis*. Il ne doit y avoir **ni avertissement ni erreur** à cette étape.

1.5 Simulation fonctionnelle

Déterminer les stimuli à appliquer aux entrées `a` et `b` et donner les valeurs attendues sur la sortie `s`. Réaliser ensuite la simulation **fonctionnelle** (*Functional*) avec le *Simulator Tool*, en cliquant sur *Generate Functional Simulation Netlist*, pour s'assurer que l'étape de conception était correcte.



1.6 Placement et routage

L'étape de placement et routage commence par l'affectation des entrées / sorties du design aux broches du FPGA, et donc au choix des périphériques associés sur la carte. Pour cela, compléter le tableau suivant, faire l'assignation des pins (*Assignment > Pins*) puis cliquer sur *Fitter (Place & Route)*. Seuls **deux avertissements** sont autorisés ici, l'un lié à *LogicLock* et l'autre aux capacités de sortie.

Nom de l'entrée / sortie du design	Type (entrée **ou** sortie)	Périphérique de la carte de développement	Broche du FPGA
`a`			
`b`			
`s`			

1.7 Génération du bitstream

Générer le bitstream qui servira à configurer le FPGA en cliquant sur *Assembler (Generate Programming Files)*.

1.8 Configuration du FPGA

Configurer le FPGA avec le *Programmer* puis valider le bon fonctionnement de votre design en agissant sur et en observant les périphériques de la carte préalablement sélectionnés.

Évaluation À partir des rapports de placement/routage et de l'analyse de *timing*, donnez le nombre de LUT occupées par l'implantation du circuit et sa fréquence maximale de fonctionnement.

Ressources logiques combinatoires (LUT)	
Fréquence maximale de fonctionnement [MHz]	

Exercice 2 : Vote majoritaire

Spécifications Concevoir un boîtier de vote pour une association. Trois personnes votent : la présidente, la secrétaire et le trésorier. Si la majorité est atteinte, on allumera la LED tout à droite. Dans le cas contraire, on allumera la LED tout à gauche.

Conception et vérification Suivre les étapes 2 à 9 décrites précédemment pour concevoir le boitie de vote et vérifier son bon fonctionnement en simulation puis sur la carte de développement.

Évaluation À partir des rapports de placement/routage et de l'analyse de *timing*, donnez le nombre de LUT occupées par l'implantation du circuit et sa fréquence maximale de fonctionnement.

Exercice 3 : Détecteur de dépassement de niveau

Spécifications Concevoir un détecteur de dépassement de niveau qui prend en entrée un nombre entier non signé sur 4 bits représentant le niveau de liquide dans une cuve et qui détecte, pour des raisons de sûreté de fonctionnement, si ce niveau est supérieur strictement à 10. Si c'est le cas, la sortie indiquant un dépassement passe à `1`.

Conception et vérification Suivre les étapes 2 à 9 décrites précédemment pour concevoir le détecteur de dépassement de niveau et vérifier son bon fonctionnement en simulation puis sur la carte de développement.

Évaluation À partir des rapports de placement/routage et de l'analyse de *timing*, donnez le nombre de LUT occupées par l'implantation du circuit et sa fréquence maximale de fonctionnement.

Exercice 4 : Décodeur BCD vers 7 segments

4.1 Décodeur basique

Spécifications Concevoir un décodeur BCD vers 7 segments. Il prend en entrée un chiffre en BCD (*binary-coded decimal*), c'est à dire un chiffre entre 0 et 9 inclus représenté en binaire non signé sur 4 bits. Ses sorties commandent un afficheur 7 segments qui affiche le chiffre. Se référerer à l'énoncé du TP1 et à la documentation de la carte pour savoir comment commander l'afficheur. Si le nombre en entrée est en dehors de l'intervalle considéré, l'afficheur indiquera `E`.

Conception et vérification Suivre les étapes 2 à 9 décrites précédemment pour concevoir un décodeur BCD vers 7 segments et vérifier son bon fonctionnement en simulation puis sur la carte de développement.

Évaluation À partir des rapports de placement/routage et de l'analyse de *timing*, donnez le nombre de LUT occupées par l'implantation du circuit et sa fréquence maximale de fonctionnement.

4.2 Conception structurelle : mode économie d'énergie

Spécifications On souhaite ajouter une entrée `ena` (pour *enable*) au circuit précédent. Si cette entrée est à `1`, alors le décodeur fonctionne comme précédemment. Si cette entrée est à `0`, alors l'afficheur 7 segments est éteint. Utiliser un multiplexeur pour cela.

Design & Reuse

Réutiliser les composants déjà conçus et testés !

Conception et vérification Suivre les étapes 2 à 9 décrites précédemment pour concevoir un décodeur BCD vers 7 segments avec mode économie d'énergie et vérifier son bon fonctionnement en simulation puis sur la carte de développement.

Évaluation À partir des rapports de placement/routage et de l'analyse de *timing*, donnez le nombre de LUT occupées par l'implantation du circuit et sa fréquence maximale de fonctionnement.

Exercice 5 : Additionneur/soustracteur sur 4 bits

5.1 Additionneur complet

Spécifications Concevoir un additionneur complet, c'est à dire un circuit faisant la somme de ses trois entrées `a`, `b` et $\texttt{r}_{\texttt{in}}$ et dont les sorties sont `s` et $\texttt{r}_{\texttt{out}}$.

Conception et vérification Suivre les étapes 2 à 9 décrites précédemment pour concevoir un additionneur complet et vérifier son bon fonctionnement en simulation puis sur la carte de développement.

Évaluation À partir des rapports de placement/routage et de l'analyse de *timing*, donnez le nombre de LUT occupées par l'implantation du circuit et sa fréquence maximale de fonctionnement.

5.2 Conception structurelle d'un additionneur/soustracteur sur 4 bits

Spécifications Concevoir un additionneur/soustracteur sur 4 bits, c'est à dire un circuit numérique réalisant la somme ou la différence de deux nombres sur 4 bits et dont le résultat sera sur 4 bits également. Inclure les sorties associées aux quatre drapeaux indicateurs d'état.

Conception et vérification Suivre les étapes 2 à 9 décrites précédemment pour concevoir un additionneur/soustracteur sur 4 bits et vérifier son bon fonctionnement en simulation puis sur la carte de développement.

Évaluation À partir des rapports de placement/routage et de l'analyse de *timing*, donnez le nombre de LUT occupées par l'implantation du circuit et sa fréquence maximale de fonctionnement.

Exercice 6 : Calculatrice (ou presque)

6.1 $a + b = s$ **pour** $s \in [0, 9]$

Spécifications Concevoir une calculatrice capable d'ajouter deux nombres **non signés** sur 4 bits, dont la somme est comprise entre 0 et 9, afin de pouvoir afficher le résultat sur un seul afficheur 7 segments. Si le résultat est en dehors de l'intervalle spécifié, un `E` sera affiché. Afficher également les deux opérandes.

Conception et vérification Suivre les étapes 2 à 9 décrites précédemment pour concevoir cette calculatrice et vérifier son bon fonctionnement en simulation puis sur la carte de développement.

Évaluation À partir des rapports de placement/routage et de l'analyse de *timing*, donnez le nombre de LUT occupées par l'implantation du circuit et sa fréquence maximale de fonctionnement.

6.2 $a \pm b = s$ **pour** $s \in [-8, 7]$

Spécifications En vous inspirant de la calculatrice créée précédemment, concevoir une calculatrice permettant d'additionner ou soustraire deux nombres **signés** sur 4 bits. Pour cela, il sera nécessaire de modifier le décodeur BCD vers 7 segments afin qu'il gère **deux** afficheurs 7 segments :

- le premier affichera si nécessaire le signe $-$
- le second affichera la valeur absolue.

Conception et vérification Suivre les étapes 2 à 9 décrites précédemment pour concevoir cette calculatrice et vérifier son bon fonctionnement en simulation puis sur la carte de développement.

Évaluation À partir des rapports de placement/routage et de l'analyse de *timing*, donnez le nombre de LUT occupées par l'implantation du circuit et sa fréquence maximale de fonctionnement.

6.3 $a \pm b = s$ **pour** $s \in [-8, 15]$

Spécifications En vous inspirant de la calculatrice créée précédemment, concevoir une calculatrice permettant d'additionner ou soustraire deux nombres **signés ou non signés** sur 4 bits. Ajouter pour cela une entrée permettant de spécifier si les nombres fournis sont signés ou non. Modifier également le décodeur afin qu'il gère **trois** afficheurs 7 segments :

- le premier affichera si nécessaire le signe $-$
- les deux autres afficheront la valeur absolue.

Conception et vérification Suivre les étapes 2 à 9 décrites précédemment pour concevoir cette calculatrice et vérifier son bon fonctionnement en simulation puis sur la carte de développement.

Évaluation À partir des rapports de placement/routage et de l'analyse de *timing*, donnez le nombre de LUT occupées par l'implantation du circuit et sa fréquence maximale de fonctionnement.