# Sujet n°7 - Compteur/Décompteur d'Événements sur FPGA

**Sujet n°7 - Compteur/Décompteur d'Événements sur FPGA....1**
1. Introduction....2
2. Cahier des charges....3
a) Objectif....3
b) Spécifications Fonctionnelles....3
c) Spécifications Techniques....3
d) Décomposition Modulaire....3
3. Décomposition de la problématique en fonction élémentaire....4
4. Schéma bloc....5
5. Analyse, codage et validation de chaque fonction élémentaire....6
a) Comptage / Décomptage / Reset....6
b) Détection de dépassement....6
c) Affichage....6
d) BCD-7Seg....7
6. Explications pour chaque modules....8
a) Module de Comptage/Décomptage/Reset....8
Programmation du module CompteurDecompteurReset....9
Simulation du module CompteurDecompteurReset....9
b) Module d'Affichage....10
Programmation du module Affichage....10
Simulation du module Affichage....10
c) Module de Détection de Dépassement....11
Programmation du module Détection de Dépassement....11
Simulation du module Détection de Dépassement....11
d) Module BCD vers 7 Segments (BCD-7Seg)....12
Programmation du module Détection de Dépassement....12
Simulation du module Détection de Dépassement....12
7. Conclusion....13

# 1. Introduction

Dans le cadre de ce projet, nous avons pour objectif de concevoir et de réaliser un **compteur/décompteur d'événements** sur une carte FPGA **Cyclone IV DE2-115**. Ce projet permettra de mettre en pratique les compétences en programmation et de synthèse matérielle sur FPGA vu lors des précédents TPs, tout en respectant des contraintes spécifiques définies dans un cahier des charges technique.

Le système à développer doit être capable de compter ou de décompter des événements, simulés par la pression d'un bouton poussoir, avec un affichage numérique sur trois afficheurs 7 segments. Il s'agit également de gérer des dépassements de valeurs, de limiter le compteur entre 0 et 999, et d'allumer une LED rouge en cas de dépassement des bornes. Une fonctionnalité de remise à zéro doit aussi être implémentée pour réinitialiser le compteur.

Ce rapport décrit le processus complet de développement, depuis l'analyse du cahier des charges jusqu'à l'intégration et la validation du système sur la carte FPGA. Nous y détaillerons la décomposition du problème en sous-fonctions élémentaires, leur codage, ainsi que les tests et validations permettant d'assurer le bon fonctionnement du système final.

# 2. Cahier des charges

## a) Objectif

Concevoir un **compteur/décompteur d'événements** de 0 à 999, affiché sur 3 afficheurs 7 segments, avec gestion des limites et des dépassements, sur une carte FPGA **Cyclone IV (DE2-115)**.

## b) Spécifications Fonctionnelles

- **Comptage/Décomptage** :
  - Incrémenter/décrémenter via un **bouton poussoir (BP)**.
  - Sélection du mode via un **interrupteur** (comptage si haut, décomptage si bas).
  - **Blocage** à 999 en comptage et à 0 en décomptage, avec allumage d'une **LED rouge** en cas de dépassement.
- **Remise à zéro (RAZ)** :
  - Réinitialisation du compteur à **0** via un BP dédié. La **LED rouge** s'éteint.

- **Affichage** :
  - Affichage de la valeur du compteur sur **3 afficheurs à 7 segments** (de 000 à 999).

## c) Spécifications Techniques

- **Anti-rebond** pour les BP afin d'éviter les déclenchements intempestifs.
- **Limitation des valeurs** à 0 et 999, avec blocage et indication par **LED rouge** en cas de dépassement.
- **Interface utilisateur** :
  - 2 **BP** : un pour l'incrément/décrément, un pour la remise à zéro.
  - 1 **interrupteur** pour le mode comptage/décomptage.
  - 3 **Afficheurs 7 segments** pour l'affichage du compteur.

## d) Décomposition Modulaire

- **Module de comptage/décomptage** avec gestion des limites.
- **Module d'affichage** pour les 7 segments.
- **Détection de dépassement** avec gestion de la **LED rouge**.
- **Remise à zéro (RAZ)** avec extinction de la LED.

## 3. Décomposition de la problématique en fonction élémentaire

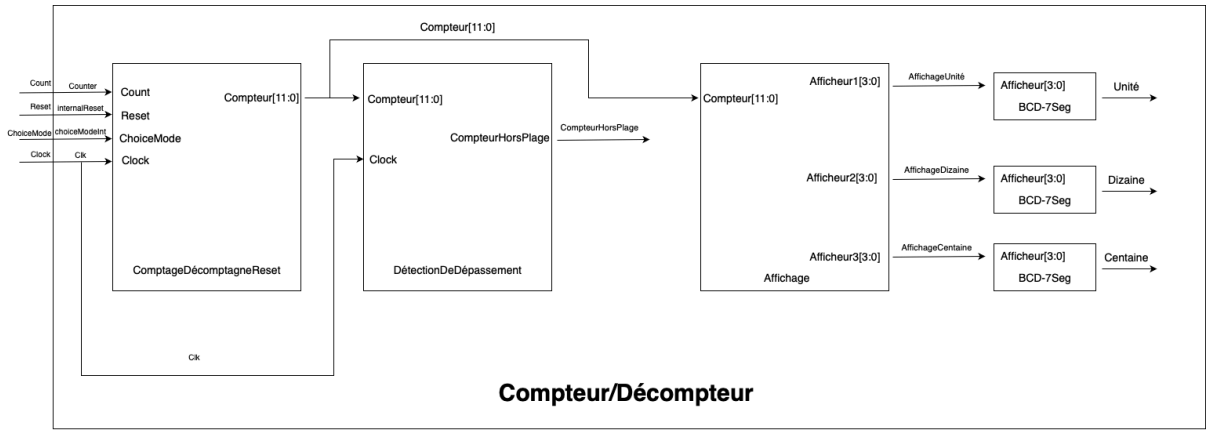
| Type de module élémentaire | Rôle | Interface | Spécification, contraintes, remarque |
|---|---|---|---|
| Comptage / Décomptage / Reset | Incrémenter ou décrémenter les valeurs (Comprise entre 0 et 999) & remise à zéro | E : Count<br>E : ChoiceMode<br>E : Reset<br>S : Compteur[11:0] | Incrémenter et décrémenter de -2048 à 2047 le compteur et permet de remettre à zéro le compteur via l'interrupteur reset |
| Affichage | Afficher sur les 3 afficheurs les nombres | E : Compteur[11:0]<br>S : Afficheur1[4:0]<br>S : Afficheur2[4:0]<br>S : Afficheur3[4:0] | Permet d'afficher notre valeur de compteur sur 3 afficheurs 7 segments. La valeur maximum d'affichage est 999 et le minimum 0. |
| Détection de dépassement | Lorsque la valeur est inférieur à 0 ou supérieur à 999 + 1 alors la LED s'allume en rouge | E : Compteur[11:0]<br>S : Compteur[11:0]<br>S : CompteurHorsPlage | Permet de détecter si notre compteur est hors plage. Si nous sommes hors plage alors une LED s'allume en rouge |

## 4. Schéma bloc

Dans le cadre de ce projet de **compteur/décompteur d'événements sur FPGA**, le schéma bloc va représenter les modules essentiels tels que le **module de comptage/décomptage**, le **module d'affichage sur les 7 segments**, et le **module de détection de dépassement**.

Chaque module sera associé à des entrées et sorties spécifiques, facilitant ainsi la compréhension de la communication entre les différentes parties du système. Cette approche modulaire permettra de mieux organiser la conception et la mise en œuvre du système sur la carte FPGA, tout en respectant les contraintes définies dans le cahier des charges.

Le schéma bloc décrit également la manière dont les boutons poussoirs, l'interrupteur, les afficheurs 7 segments, et les LEDs sont interconnectés pour former un système cohérent et fonctionnel.



# 5. Analyse, codage et validation de chaque fonction élémentaire

### a) Comptage / Décomptage / Reset

```
A chaque appuie sur front descendant sur count ou front montant Reset alors alors
        vérifier si Reset =1 alors
                compteur = 0
        sinon
                Vérifier si ChoiceMode = 1 (Incremente) alors
                        compteur = compteur +1
                sinon (ChoiceMode = 0 (decremente))
                        compteur = compteur - 1
                finsi
        finsi
FIN
```

### b) Détection de dépassement

```
A chaque changement de Compteur
        Vérifier Si Compteur <=999 ET Compteur >= 0 alors
                LedRouge = 0
        sinon
                LedRouge = 1
        finsi
FIN
```

### c) Affichage

```
A chaque changement de Compteur
        Afficheur1 = unitéCompteur
        Afficheur2 = dizaineCompteur
        Afficheur3 = centaineCompteur
FIN
```

### d) BCD-7Seg

```
A chaque changement de BinIn faire
        switch (BinIn)
                cas 0 : ISegOut = 1111110
                cas 1 : ISegOut = 0110000
                cas 2 : ISegOut = 1101101
                cas 3 : ISegOut = 1111001
                cas 4 : ISegOut = 0110011
                cas 5 : ISegOut = 1011011
                cas 6 : ISegOut = 1011111
                cas 7 : ISegOut = 1110000
                cas 8 : ISegOut = 1111111
                cas 9 : ISegOut = 1111011
                cas defaut : ISegOut = 0000000
        FinSwitch
Inversion binaire de ISegOut
Fin
```

# 6. Explications pour chaque modules

Dans cette partie, nous allons détailler le rôle et le fonctionnement de chaque module de notre système de **compteur/décompteur d'événements sur FPGA**. Chaque module joue un rôle spécifique dans la gestion du comptage, de l'affichage et de la détection des dépassements.

Il est important de noter que le système continue de compter au-delà des limites imposées, même si ces valeurs ne peuvent pas être affichées. L'afficheur montrera 0 lorsque le compteur est en dessous de la limite, et 999 lorsqu'il dépasse la limite supérieure. Bien que le compteur puisse réellement compter de -2048 à 2047 ($2^{12}$ signés), l'affichage reste restreint aux valeurs comprises entre 0 et 999, comme mentionné précédemment.

## a) Module de Comptage/Décomptage/Reset

Le **module de comptage/décomptage** est au cœur du système, responsable de l'incrémentation ou de la décrémentation de la valeur du compteur en fonction des interactions de l'utilisateur. Il prend en compte les entrées suivantes :

- **Count** : un bouton poussoir pour incrémenter ou décrémenter la valeur du compteur.
- **ChoiceMode** : un interrupteur pour sélectionner le mode comptage ou décomptage (comptage si haut, décomptage si bas).
- **Reset** : un interrupteur pour réinitialiser le compteur à zéro.

Ce module gère également les limites de comptage, c'est-à-dire :

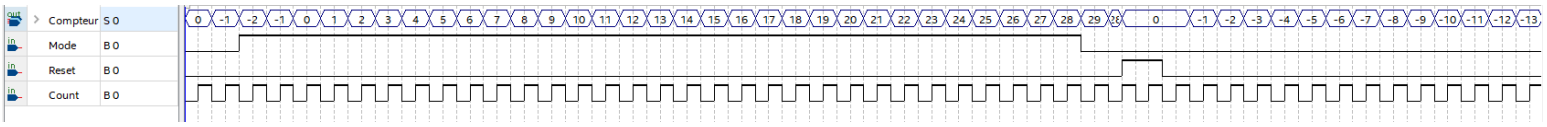
- En mode comptage, la valeur maximale est de 999. Si le compteur atteint cette limite, une LED sera allumée pour indiquer que la limite à était atteinte.
- En mode décomptage, la valeur minimale est de 0. Si le compteur atteint 0,une LED sera allumée pour indiquer que la limite à était atteinte.

Si le bouton de remise à zéro est activé, le compteur est réinitialisé à 0 et par la même occasion la LED qui indique le dépassement s'éteindra.

## Programmation du module CompteurDecompteurReset

```
1  module CompteurDecompteurReset(Count, Mode, Reset, Compteur);
2
3  input wire Count;
4  input wire Mode;
5  input wire Reset;
6  output wire signed [11:0]Compteur;
7
8  reg signed [11:0] InternalCompteur;
9
10 always @(negedge Count or posedge Reset)
11 begin
12    if (Reset == 1)
13    begin
14       InternalCompteur <= 0;
15    end else
16    begin
17       if (Mode == 1)
18          begin
19             InternalCompteur <= InternalCompteur + 1;
20          end
21       else
22          begin
23             InternalCompteur <= InternalCompteur - 1;
24          end
25    end
26 end
27
28 assign Compteur = InternalCompteur;
29
```

## Simulation du module CompteurDecompteurReset




Nous avons simulé le module de la manière suivante : dans un premier temps, nous avons sélectionné le mode décomptage en réglant le Mode à 0. Comme attendu, le compteur se décrémente à chaque front descendant du signal provenant du bouton poussoir **Count**. Ensuite, en passant le Mode à 1, le système bascule en mode comptage, et le compteur s'incrémente correctement. Enfin, lorsque le signal **Reset** passe à 1 sur un front montant, le compteur se réinitialise à 0, conformément au comportement attendu.

### b) Module d'Affichage

Le **module d'affichage** est chargé d'indiquer la valeur du compteur sur trois afficheurs à 7 segments. Il prend en entrée la valeur du compteur, qui est divisée en unités, dizaines et centaines pour l'affichage.

- **Entrée** : la valeur du compteur sur 12 bits (Compteur[11:0]), représentant une valeur entre 0 et 999.
- **Sorties** : trois valeurs en BCD correspondant aux chiffres des unités, dizaines et centaines, envoyées aux afficheurs 7 segments.

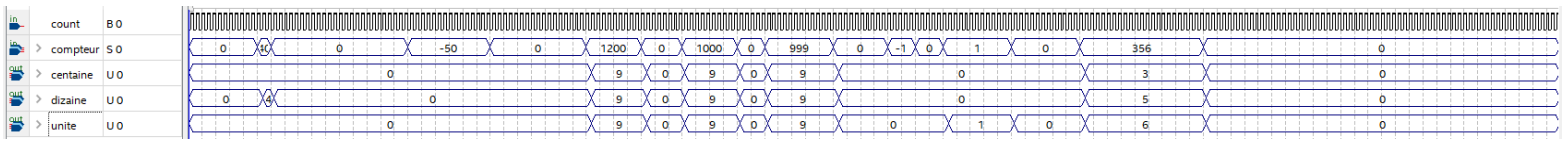
#### Programmation du module Affichage

```
module Affichage(
    input wire count,              // Signal d'horloge
    input signed [11:0]compteur,        // Compteur
    output reg [3:0] unite,      // Chiffre des unités
    output reg [3:0] dizaine,    // Chiffre des dizaines
    output reg [3:0] centaine    // Chiffre des centaines
);

 always @(posedge count) begin
   if (compteur <= 999 && compteur >= 0) begin
      unite    = compteur % 10;
      dizaine  = (compteur / 10) % 10;
      centaine = (compteur / 100);
   end
   else begin
      if (compteur < 0)
         begin
            unite    = 0;
            dizaine  = 0;
            centaine = 0;
        end
        else begin
            unite    = 9;
            dizaine  = 9;
            centaine = 9;
        end
     end
 end

 endmodule
```

#### Simulation du module Affichage




Dans cette simulation, nous avons injecté différentes valeurs au compteur pour effectuer des tests. On constate que pour les valeurs comprises entre 0 et 999, les chiffres des centaines, dizaines et unités sont correctement affichés. Lorsque les valeurs dépassent les limites imposées, le système fonctionne comme prévu : il affiche 0 lorsque la valeur est inférieure à la limite, et 999 lorsque la valeur dépasse la limite supérieure.

### c) Module de Détection de Dépassement

Le **module de détection de dépassement** permet de gérer les situations où la valeur du compteur sort de la plage autorisée (entre 0 et 999). Il détecte si la valeur est en dehors de ces bornes et active une LED rouge pour indiquer un dépassement.

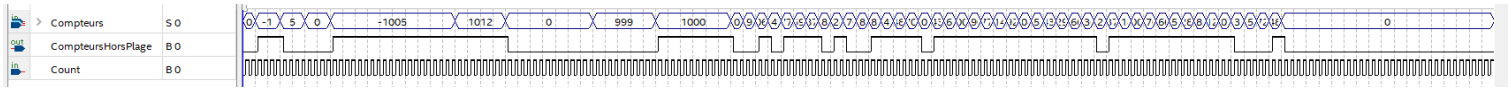
- **Entrée** : la valeur du compteur sur 12 bits.
- **Sortie** : une LED qui s'allume si la valeur dépasse 999 en mode comptage ou descend en dessous de 0 en mode décomptage.

Ainsi, si une valeur hors plage est atteinte, la LED s'allume pour signaler qu'une limite à était atteinte.

#### Programmation du module Détection de Dépassement

```
1   module DetectionDeDepassement(Count, Compteurs, CompteursHorsPlage);
2
3   input wire Count;
4   input wire signed[11:0]Compteurs;
5   output reg CompteursHorsPlage;
6
7   always @(posedge Count)
8   begin
9       if (Compteurs < 0 || Compteurs > 999)
10          CompteursHorsPlage <= 1; // Indique que le compteur est hors plage
11      else
12          CompteursHorsPlage <= 0; // Indique que le compteur est dans la plage
13  end
14
15  endmodule
```

#### Simulation du module Détection de Dépassement




Comme le montre la simulation, lorsque le compteur est à 0, il reste dans la plage autorisée. Cependant, dès que la valeur du compteur devient -1 ou inférieure, le système considère que nous sommes hors plage. De plus, tant que le compteur reste compris entre 0 et 999, il est dans la plage valide. En revanche, si la valeur du compteur atteint ou dépasse 1000, le système signale également un dépassement et nous sommes alors hors plage.

### d) Module BCD vers 7 Segments (BCD-7Seg)

Ce module est responsable de la conversion des valeurs BCD générées par le module d'affichage en signaux contrôlant les afficheurs à 7 segments. Il traduit chaque chiffre décimal en un motif spécifique de segments à allumer pour afficher correctement les chiffres sur les afficheurs.

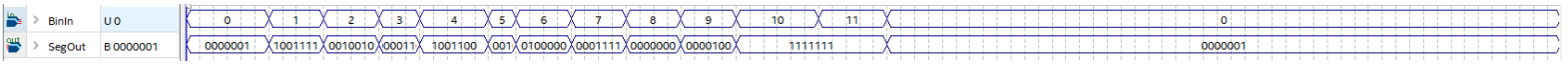
- **Entrée** : une valeur BCD de 4 bits (0 à 9) correspondant à un chiffre du compteur.
- **Sortie** : un ensemble de 7 signaux pour contrôler les segments de l'afficheur, permettant d'afficher le chiffre correspondant.

Ce module est indispensable pour que les chiffres du compteur soient correctement affichés sous forme lisible pour l'utilisateur.

#### Programmation du module Détection de Dépassement

```
1   module BCD7Seg(BinIn, SegOut);
2
3   input[3:0] BinIn;          // Binary input value
4
5
6   output[0:6] SegOut;
7
8   reg[0:6] iSegOut;       // internal seg value (positive logic / CC)
9
10  assign SegOut[0:6] = ~iSegOut[0:6]; // Adapt to CA display
11
12
13  always @(BinIn)
14  begin
15     case (BinIn)
16                    // abcdefg
17        0 : iSegOut <=    7'b1111110;
18        1 : iSegOut <=    7'b0110000;
19        2 : iSegOut <=    7'b1101101;
20        3 : iSegOut <=    7'b1111001;
21        4 : iSegOut <=    7'b0110011;
22        5 : iSegOut <=    7'b1011011;
23        6 : iSegOut <=    7'b1011111;
24        7 : iSegOut <=    7'b1110000;
25        8 : iSegOut <=    7'b1111111;
26        9 : iSegOut <=    7'b1111011;
27        default : iSegOut <=    7'b0000000;
28     endcase
29  end
30
31
32
33  endmodule
```

#### Simulation du module Détection de Dépassement




Dans cette simulation, nous vérifions que pour chaque valeur décimale donnée en entrée à **BinIn**, l'afficheur 7 segments **SegOut** génère correctement le motif binaire correspondant. Cela permet d'activer les segments appropriés afin d'afficher le chiffre correspondant à la valeur binaire d'entrée.

# 7. Conclusion

Nous avons atteint les objectifs fixés pour ce projet en concevant et en réalisant avec succès un compteur/décompteur d'événements. Le système répond aux exigences spécifiées dans le cahier des charges, notamment en matière de comptage, décomptage, remise à zéro, et gestion des limites avec affichage numérique et signalisation grâce à une LED en cas de dépassement.

Chaque module a été développé, testé et validé, assurant ainsi le bon fonctionnement global du système. Le compteur fonctionne correctement pour des valeurs comprises entre 0 et 999, avec un comportement conforme aux attentes lorsque les limites sont atteintes ou dépassées. Bien que le système continue de compter au-delà de ces bornes, l'afficheur est limité à afficher 0 pour les valeurs inférieures et 999 pour les valeurs supérieures. De plus, la gestion des segments pour l'affichage et la détection des dépassements a été implémentée avec succès.

En conclusion, ce projet a permis de mettre en œuvre nos compétences en conception matérielle et en synthèse sur FPGA, en respectant les contraintes imposées. Le système final est fonctionnel et stable, démontrant une bonne maîtrise des aspects techniques étudiés durant les travaux pratiques.