

TP 3

Conception des machines séquentielles en utilisant « Altera Finite State Machine »


CHIRAZ TRABELSI
Et
ALEXANDRE BRIERE

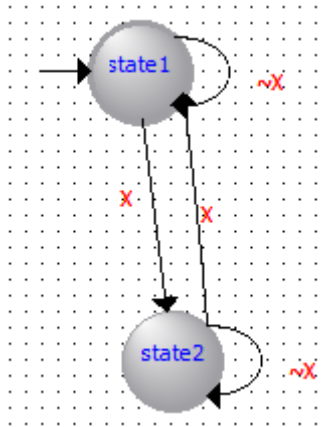


TP3 SYS2044



Exercice 1

Cet exercice vise à implémenter le système vérificateur de parité paire vu en TD1 en utilisant la saisie de machines d'état ALTERA (File → New → State Machine File). Le schéma électrique correspondant à la machine d'états saisie sera généré automatiquement par Quartus II.

Cliquer « File → New » ou  puis sélectionner «State Machine File ». Une fenêtre de l'éditeur graphique de machine à état est ouverte.





En suivant l'architecture ci-dessus :

- Créer l'entrée X en cliquant sur 
- Créer la sortie F en cliquant sur 

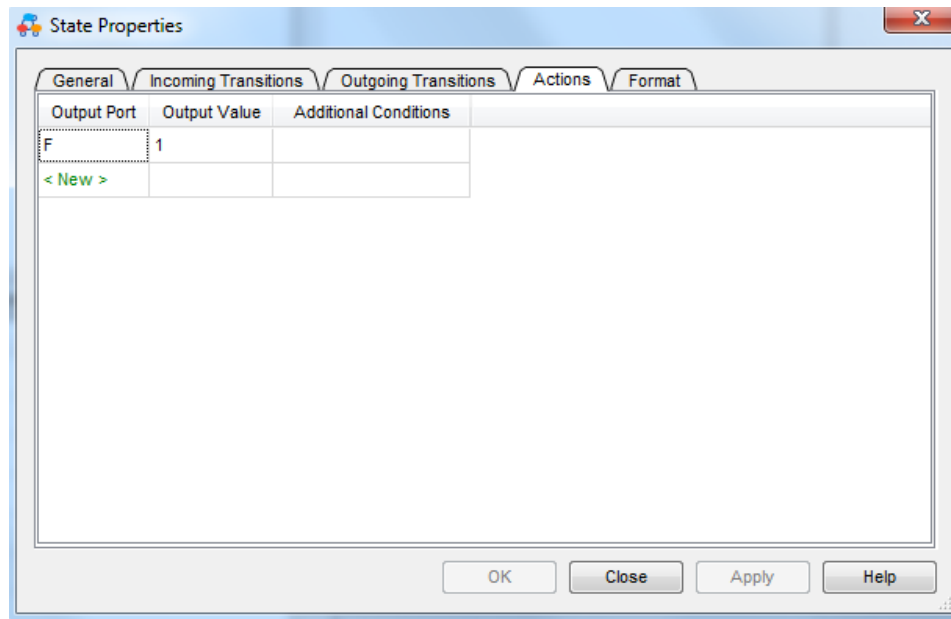
Vous obtenez le résultat suivant :


Input Table	
Input Port	
1	reset
2	clock
3	X

Output Table	
Output Port	
1	F

- Placer les 2 états de la machine en cliquant sur , l'état state1 possède par défaut une flèche sur la gauche, c'est l'état après la remise à zéro de la machine.
- Placer les liens entre les états en cliquant sur .
- Pour ajouter les conditions des transitions, double-clic sur les liens entre états permet d'éditer les conditions. Si la transition est valide pour X=1, écrire X, si la transition est valide pour X=0, écrire ~X. Le signe « ~ » représente l'opérateur « non ».
- Indiquer l'état des sorties pour chaque état de la machine. Double clic sur chaque état puis onglet «action» et ajouter la valeur de la sortie F (F=1 pour state1 et F=0 pour state2).

TP3 SYS2044



Tous les détails de la machine à états sont affichables en cliquant sur .

Génération du fichier VHDL

La machine à états est terminée, créer le fichier VHDL correspondant, en cliquant sur .

Le fichier VHDL généré est le suivant :

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

```
ENTITY SM IS
  PORT (
    reset : IN STD_LOGIC := '0';
    clock : IN STD_LOGIC;
    X : IN STD_LOGIC := '0';
    F : OUT STD_LOGIC
  );
END SM;
```

Déclaration des entrées et des sorties

```
ARCHITECTURE BEHAVIOR OF SM
IS
```

```
  TYPE type_fstate IS (state1,state2);
  SIGNAL fstate : type_fstate;
  SIGNAL reg_fstate : type_fstate;
```

Le signal fstate désigne l'état courant (qui se met à jour au front montant de l'horloge).

Le signal reg_fstate désigne l'état suivant, qui peut être vu comme une préparation de la valeur qui sera affectée à l'état courant (fstate) au prochain front d'horloge (en se basant sur la valeur de l'état courant et celle de l'entrée).

```
BEGIN
  PROCESS (clock,reg_fstate)
  BEGIN
    IF (clock='1' AND clock'event) THEN
```

TP3 SYS2044

```

fstate <= reg_fstate;
END IF;
END PROCESS;

PROCESS (fstate,reset,X)
BEGIN
  IF (reset='1') THEN
    reg_fstate <= state1;
    F <= '0';
  ELSE
    F <= '0';
    CASE fstate IS
      WHEN state1 =>
        IF ((X = '1')) THEN
          reg_fstate <= state2;
        ELSIF (NOT((X = '1')))) THEN
          reg_fstate <= state1;
        -- Inserting 'else' block to prevent latch inference
        ELSE
          reg_fstate <= state1;
        END IF;
      WHEN state2 =>
        IF ((X = '1')) THEN
          reg_fstate <= state1;
        ELSIF (NOT((X = '1')))) THEN
          reg_fstate <= state2;
        -- Inserting 'else' block to prevent latch inference
        ELSE
          reg_fstate <= state2;
        END IF;
      WHEN OTHERS =>
        F <= '0';
        F <= 'X';
        report "Reach undefined state";
    END CASE;
  END IF;
END PROCESS;
END BEHAVIOR;

```

Mise à jour des états

Effet du RESET sur l'état et la sortie

Indication, pour chaque état courant, la/les conditions de passage à un autre état et la valeur de la sortie
Exemple : Si l'état courant est state1 et que X=1, l'état suivant est state2. Si X=0, l'état suivant est state1, sinon l'état courant ne change pas.
Rq1 : L'instruction CASE ... WHEN en VHDL est l'équivalent de l'instruction switch ... case en langage C.

Rq2 : le dernier « else » est utilisé seulement parce que si ce n'est pas le cas, le logiciel va créer des bascules pour gérer ce cas, ce qui résulte en une perte de ressources (bascules). Il faut donc veiller à écrire toujours les sinon dans un programme VHDL.

Modification dans le fichier VHDL généré

Dans Quartus, les fichiers VHDL générés automatiquement à partir d'une machine à états mettent les sorties à zéro quand il y a un reset. Ce qu'on veut dans notre exemple, c'est qu'au moment de la réinitialisation (mise à l'état initial), la sortie de se trouve à 1. Il faut donc faire la modification suivante :

```

PROCESS (fstate,reset,X)
BEGIN
  IF (reset='1') THEN
    reg_fstate <= state1;

```

TP3 SYS2044

F <= '1'

Simulation RTL

Le fichier généré sera utilisé lors de la simulation, dans le cas d'un projet ayant de multiples sources il est nécessaire de faire apparaître ce fichier dans le projet (Files → add) et le mettre en Top-Level. Remarque : ici on n'est pas obligé de détacher le fichier .smf de la machine d'états même s'il a le même nom que le fichier VHDL puisque un fichier .smf ne peut pas implémenté directement à la différence d'un fichier schéma .bdf.

Désactiver l'optimisation du placement physique, compiler le projet et lancer la simulation RTL.

Vous pouvez utiliser un script de simulation qui ressemble au suivant :

```
#creation de signal horloge de periode 100ns

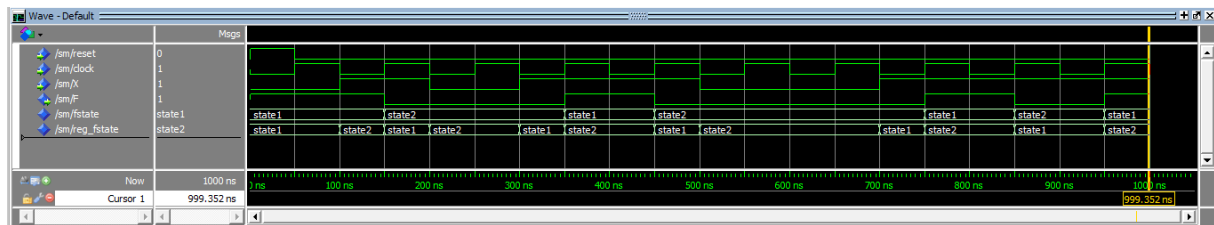
force sim:/sm/clock 0 0ns, 1 50ns -repeat 100ns

#activer le signal reset pour 50ns
force sim:/sm/reset 1 0ns, 0 50ns

#La séquence de x: 01011001, 0 au départ, 1 après 100ns (un cycle d'horloge), .....

force sim:/sm/x 0 0ns, 1 100ns, 0 200ns, 1 300ns, 1 400ns, 0 500ns, 0 600ns, 1 700ns
```

Vous êtes censés obtenir le résultat suivant :



Vérifier que cela donne bien le fonctionnement désiré :

X : 01011001

F : 10010001

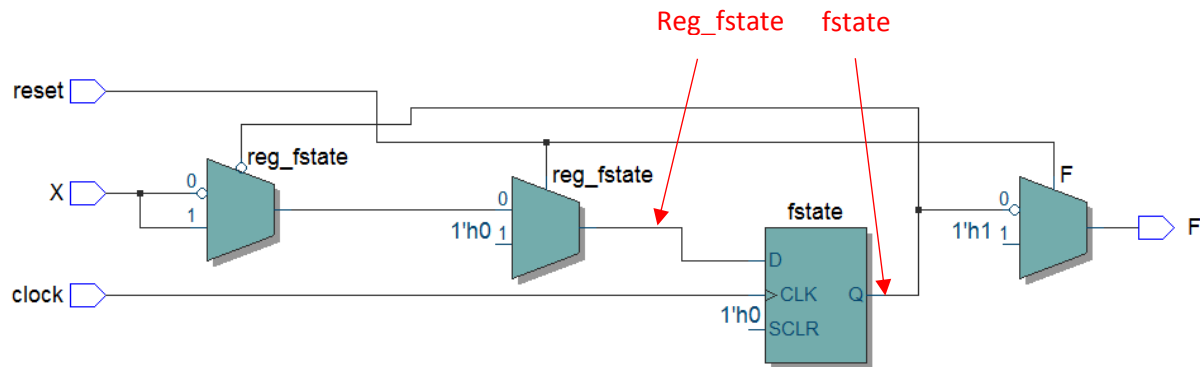
La figure montre bien que le signal fstate se met à jour au front montant de l'horloge alors que le signal reg_fstate se met à jour dès que la valeur de l'entrée change.

Appeler l'encadrant pour valider la simulation. Vous êtes censés expliquer les valeurs des signaux fstate et reg_fstate et donner leur utilité.

Structure logique de la machine à état

Cliquer sur « Tools → Netlist Viewers → RTL Viewer » permet d'afficher le schéma de la structure fonctionnelle

TP3 SYS2044



La figure montre que le logiciel a synthétisé la machine d'états en utilisant une bascule et des multiplexeurs.

Les deux premiers multiplexeurs peuvent être interprétés comme suit :

Si $\text{reset}=1$ alors $\text{reg_fstate} \leftarrow 0$ (dans le deuxième multiplexeur)

Si $\text{reset}=0$ alors reg_fstate prend la valeur de la sortie du premier multiplexeur (un xor entre Q et X)

\Rightarrow Si $\text{reset}=0$ et $Q=1$ alors $\text{reg_fstate} \leftarrow \text{Non}(X)$

Si $\text{reset}=0$ et $Q=0$ alors $\text{reg_fstate} \leftarrow X$

$\Rightarrow \text{reg_fstate} \leftarrow \text{NON}(\text{reset}). (Q \text{ XOR } X)$

Pour le troisième multiplexeur :

Si $\text{reset} = 1 \rightarrow F=1$

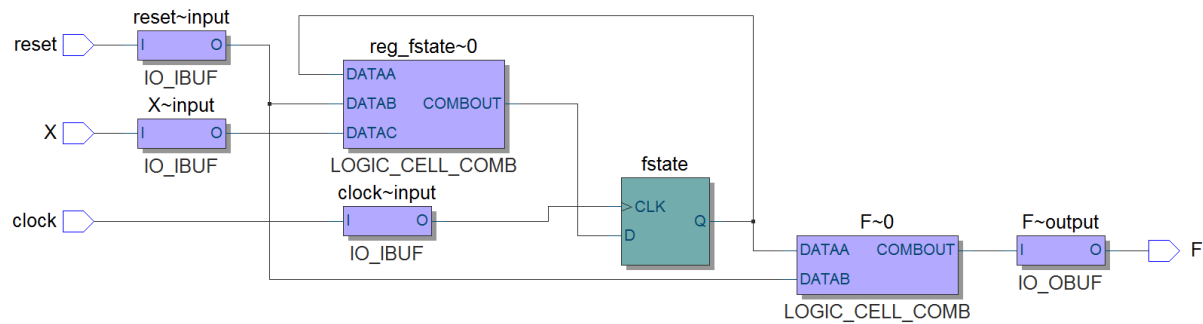
Si $\text{reset}=0 \rightarrow F= \text{NON}(Q)$

Ici on voit bien qu'on a obtenu les mêmes équations que dans le TP1 (moyennant le signal reset).

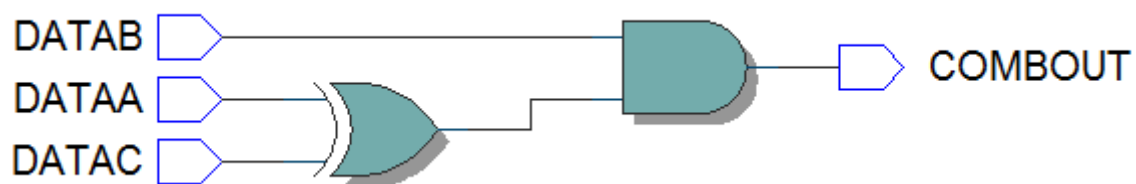
Structure synthétisée dans le FPGA de la machine à état

Cliquer « Tools \rightarrow Netlist Viewers \rightarrow Technologie Map Viewer (Post-Fitting) » permet d'afficher le schéma de la structure synthétisée (en prenant en compte les composants du FPGA cible).

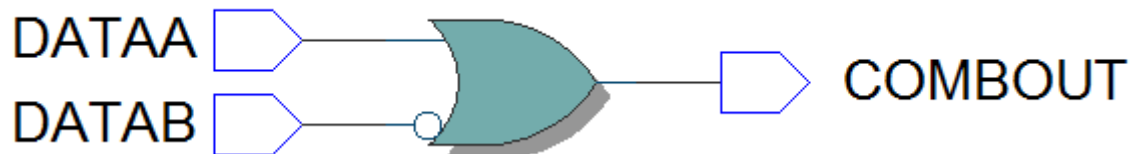
TP3 SYS2044



La structure interne du premier LOGIC_CELL peut être visualisée (clic-droit, propriétés)



Le deuxième bloc est :



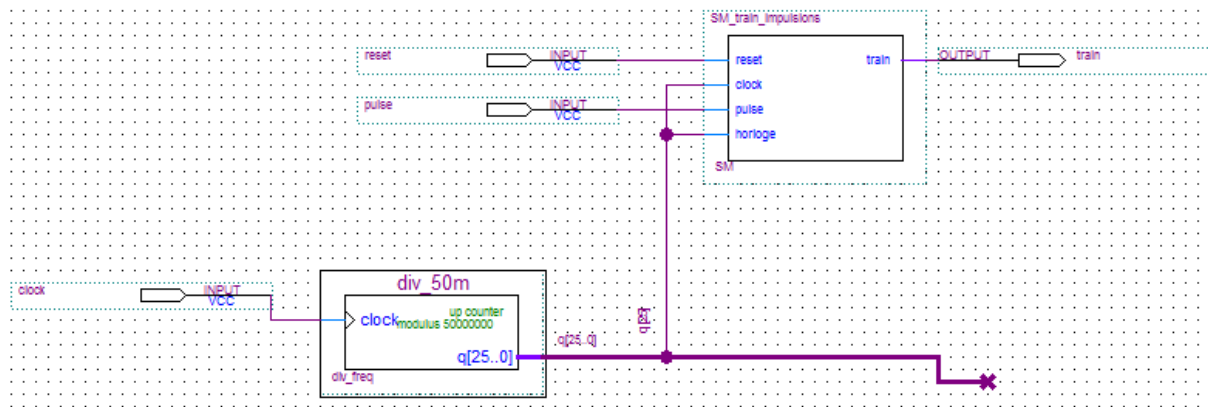
Là on voit les composants qui seront réellement implémentés dans le FPGA.

Appeler l'encadrant pour valider cette étape et expliquer le contenu des blocs obtenus.

Exercice2 : générateur de train d'impulsions

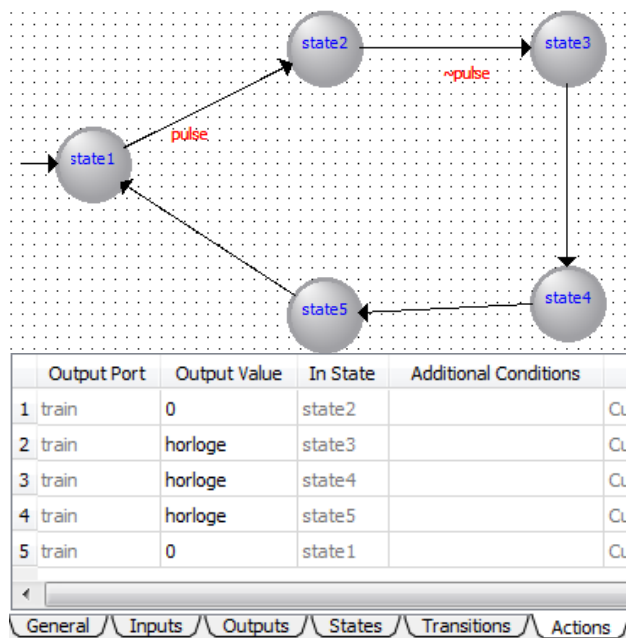
On se propose de réaliser un générateur de train de trois impulsions sur la LED LEDR0 (3 clignotement de la LED) lors de l'appui sur le switch SW1 de la carte. Le schéma final à obtenir est le suivant :

TP3 SYS2044



Dans ce schéma le symbole « SM_train_impulsions » a été généré à partir d'une machine à états qui prend en entrée la variable pulse donnée par le switch SW1 et génère trois impulsions sur la LED LEDR0 à travers la sortie train. L'entrée « reset » de la machine à états est liée au switch SW0. Le symbole « div_50m » représente un diviseur de fréquence qui prend en entrée la fréquence 50MHZ (la fréquence utilisée par la carte sur la PIN_AF14, voir le user guide de la carte) et la divise sur 50 millions pour obtenir une fréquence d'1HZ et donc une période d'horloge d'une seconde. La sortie de ce diviseur est donnée en entrée clock au composant « SM_train_impulsions ». Cette solution est utilisée pour permettre la visibilité de l'allumage des LED, sinon cela serait très rapide (en connectant l'entrée clock du « SM_train_impulsions » directement au pin PIN_AF14, on a une période d'horloge de 20ns → la LED s'allume pendant 60ns, ce qui est invisible).

La machine d'états est la suivante :



Créer la machine ci-dessus. Le signal « horloge » est une entrée à ajouter en plus de l'entrée « pulse » parce que le logiciel ne nous permet d'utiliser directement le signal « clock » qui existe par défaut dans la machine à états.

Après un front descendant sur « pulse », « l'horloge » est transmise trois fois sur la sortie train. La sortie train est donc égale au signal horloge pour les états state2, state3 et state4.

TP3 SYS2044

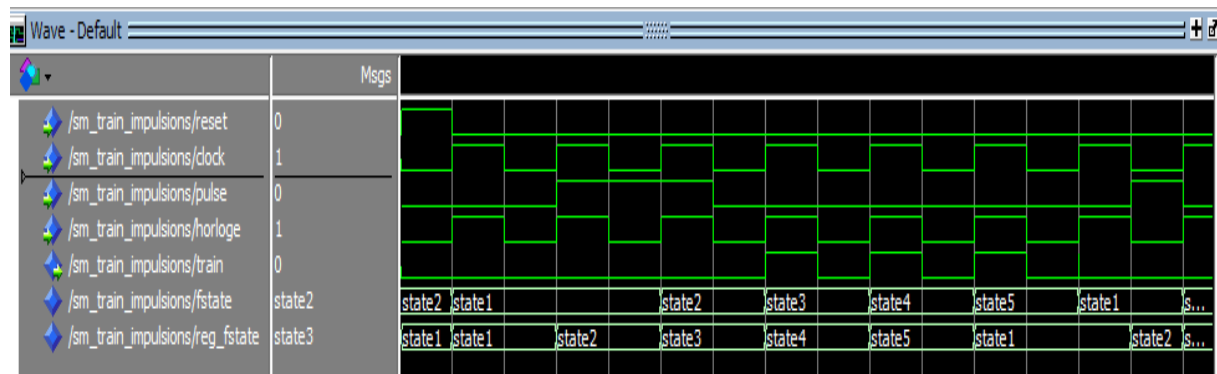
Simulation RTL de la machine d'état

Générer le fichier VHDL à partir de la machine d'états et le mettre comme Top-level
Désactiver l'optimisation du placement, compiler le projet et lancer la simulation RTL.

Lancer le script de simulation suivant :

```
force sim:/sm_train_impulsions/clock 0 0ns, 1 50ns -repeat 100ns
force sim:/sm_train_impulsions/horloge 0 0ns, 1 50ns -repeat 100ns
force sim:/sm_train_impulsions/reset 1 0ns, 0 50ns
force sim:/sm_train_impulsions/pulse 0 0ns, 1 150ns, 0 300ns, 1 700ns, 0 750ns
run 800ns
```

Vous êtes censés obtenir le résultat suivant :



-Remarquer les 3 impulsions sur la sortie train.

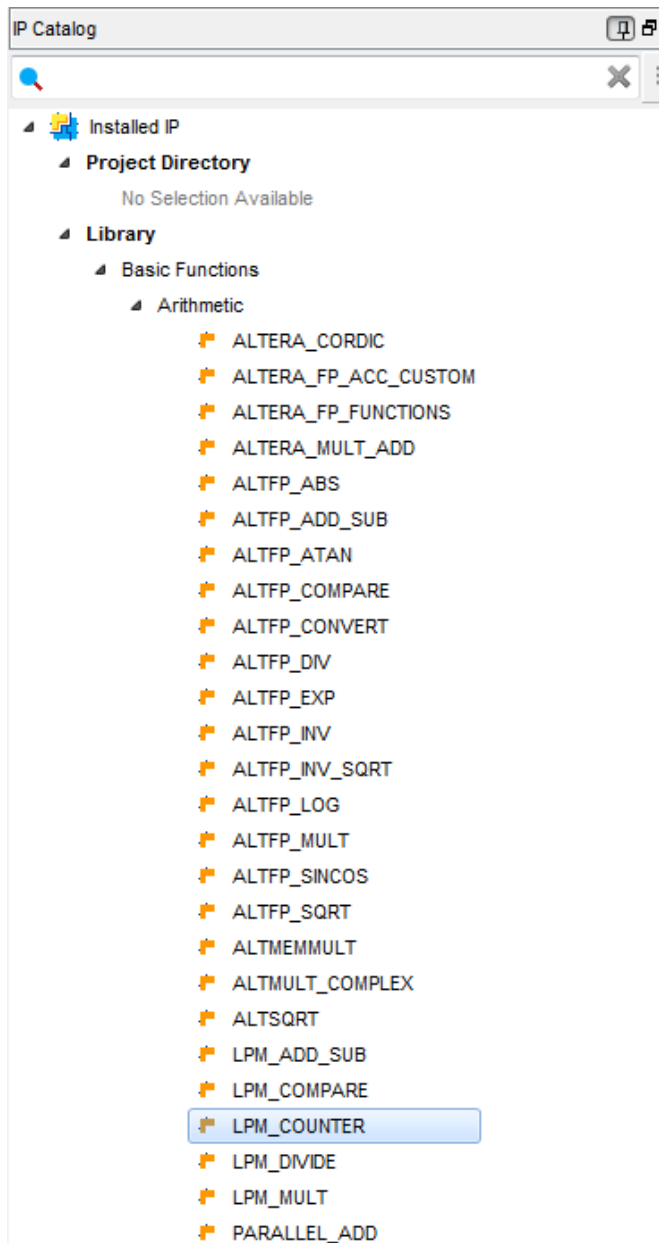
Appeler l'encadrant pour valider la simulation.

Schéma final

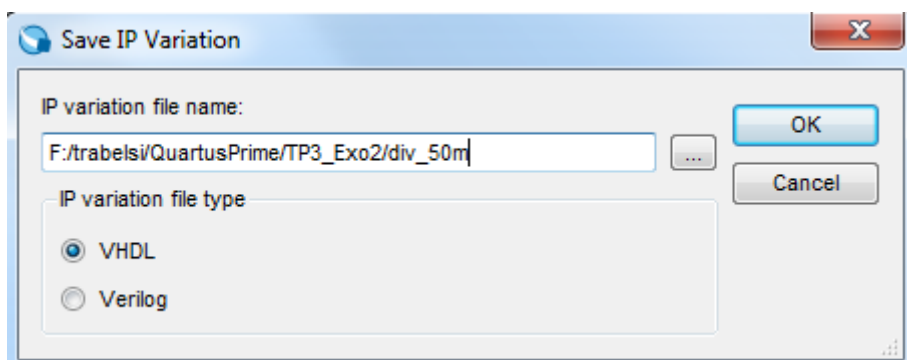
Pour le schéma final, vous allez générer un composant qui implémente la machine à état, ajouter le diviseur de fréquence et connecter les entrées/sorties. Pour cela, suivez les étapes suivantes :

- Générer un symbole à partir du fichier VHDL.
- Créer un nouveau fichier schéma et ajouter le composant généré.
- Créer un diviseur de fréquence et l'ajouter au schéma en suivant les étapes suivantes :
- Cliquer sur le menu IP Catalog à droite, sélectionner library → Basic Functions → Arithmetic → lpm_counter

TP3 SYS2044

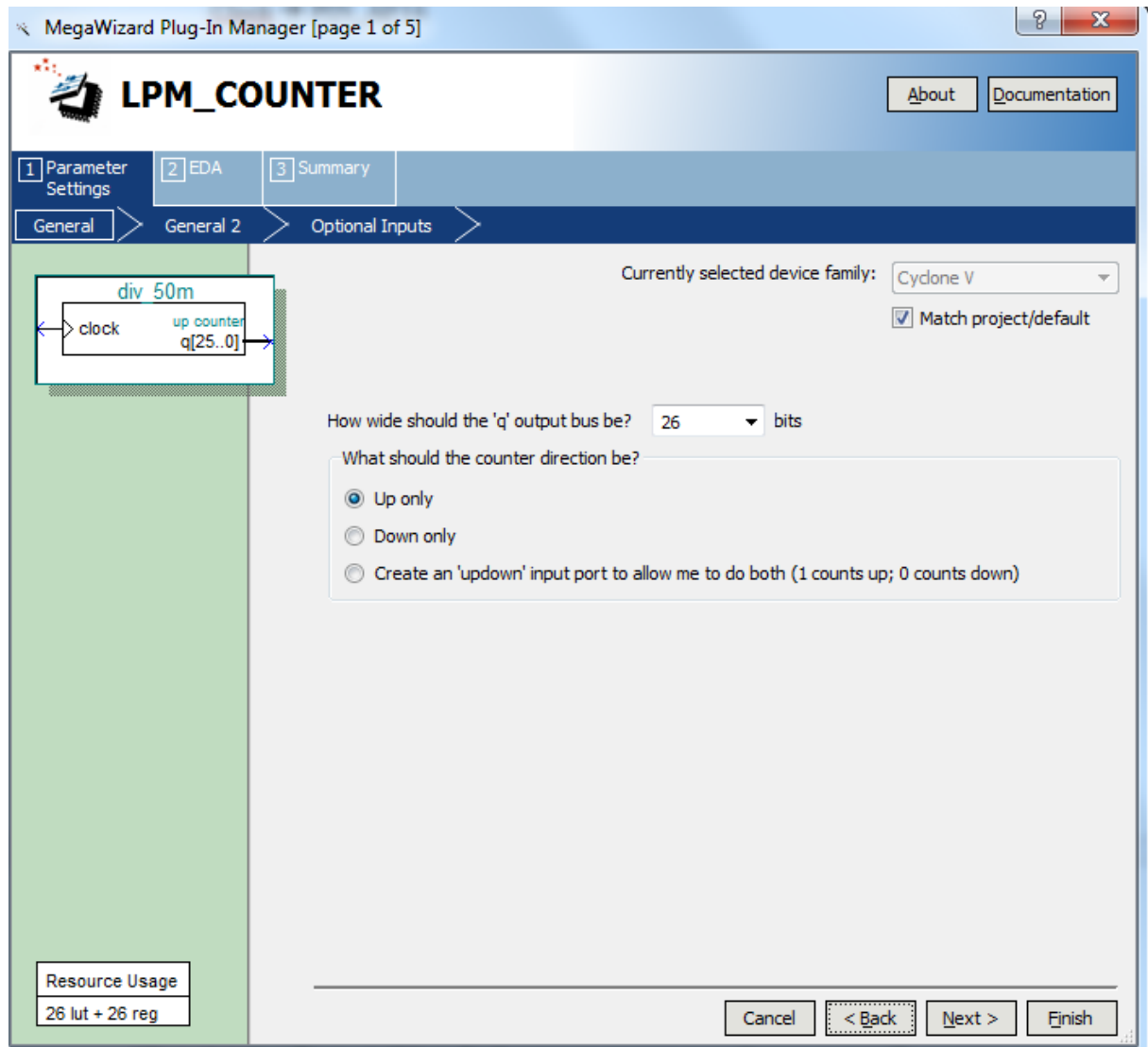


-Nommer le compteur dans a fenêtre wizard qui apparaît (exemple div_50m).



-Cliquer OK et choisir 26 bits (le compteur doit aller jusqu'à 50 millions, ce qui s'écrit sur 26 bits).

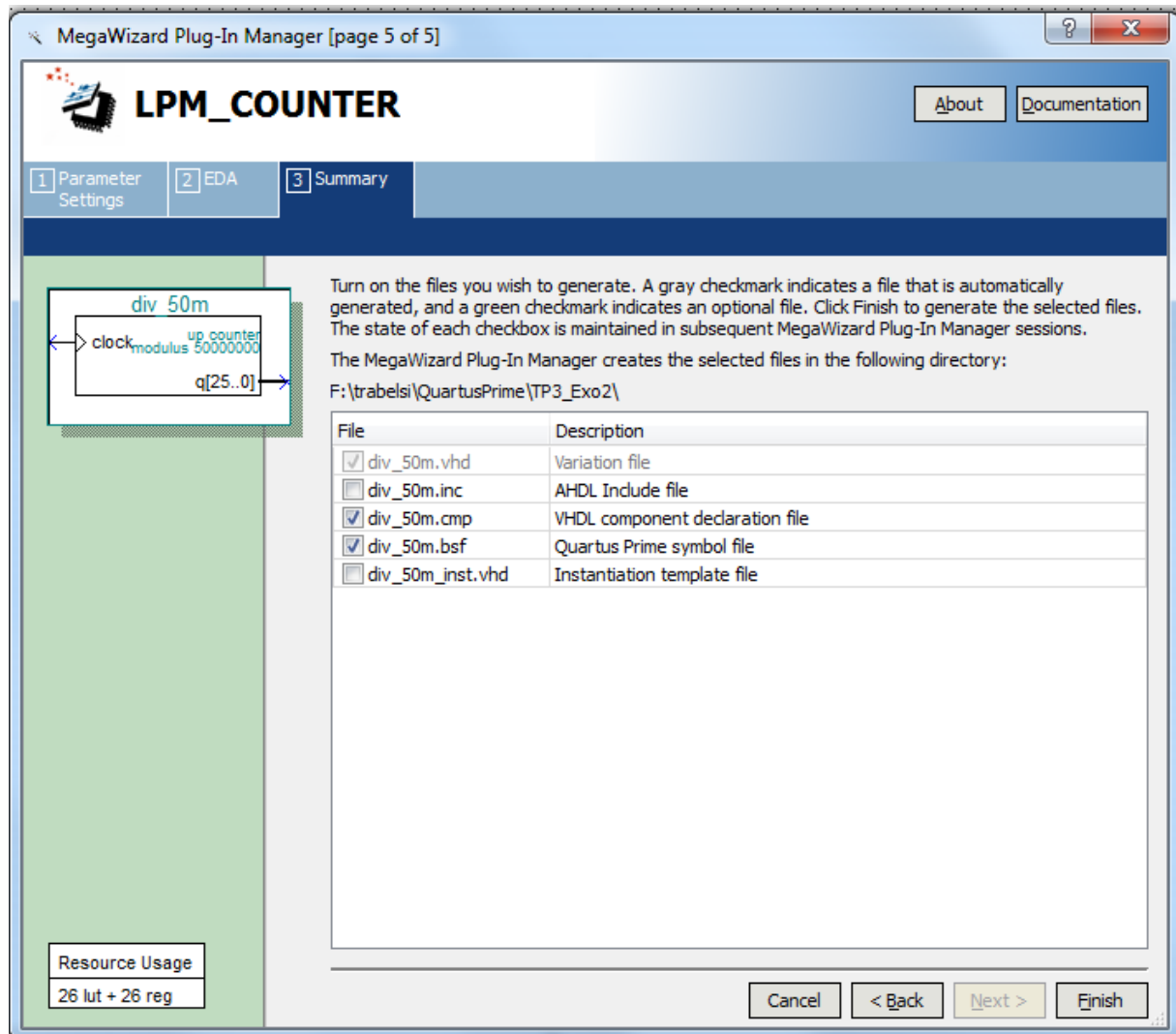
TP3 SYS2044




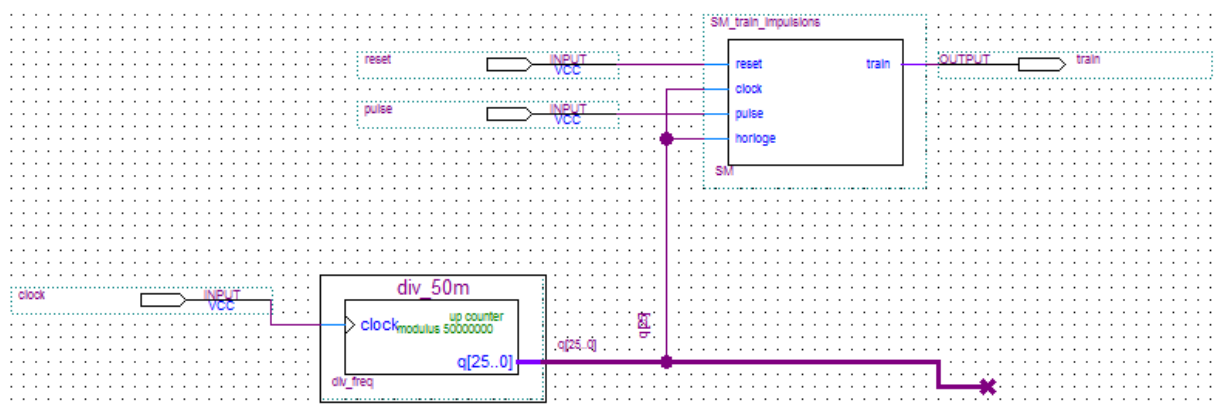
-Cliquer Next et sélectionner l'option modulus et écrire la valeur 50000000 dans le champ correspondant. Cela garantit que le compteur ne dépasse pas les 50 millions.


-Cliquer Next puis Next à la dernière page du wizard vérifier que la case « div_50m.bsf » est cochée pour pouvoir ajouter un symbole pour le composant qu'on vient de créer.

TP3 SYS2044



-Cliquer sur  et ajouter le compteur au schéma.
Faire les connexions comme indiquée dans la figure ci-dessous :



Pour la connexion entre le diviseur de fréquence et la machine à états, il faut dessiner un bus en cliquant sur , nommer le bus q[25..0] en faisant un clic-droit → Properties. Le nom ne doit pas contenir d'espaces. Lier les entrées clock et horloge de la machine d'états à ce bus. Nommer le lien q[25], ce qui fait la liaison avec le bit de poids fort du bus.

-Mettre le schéma en Top-Level et compiler.

TP3 SYS2044

-Faire l'affectation des pins en utilisant Pin Planner.

Pulse → SW[1]

Train → LEDR[0]

Reset → SW[0]

Clock → PIN_AF14

-Recompiler et tester sur la carte.

Appeler l'encadrant pour valider le résultat de l'implémentation sur la carte.

Exercice 3 : gestionnaire de carrefour

Le carrefour possède deux axes de circulation.

La voie Nord-Sud (NS) est prioritaire et au vert par défaut.

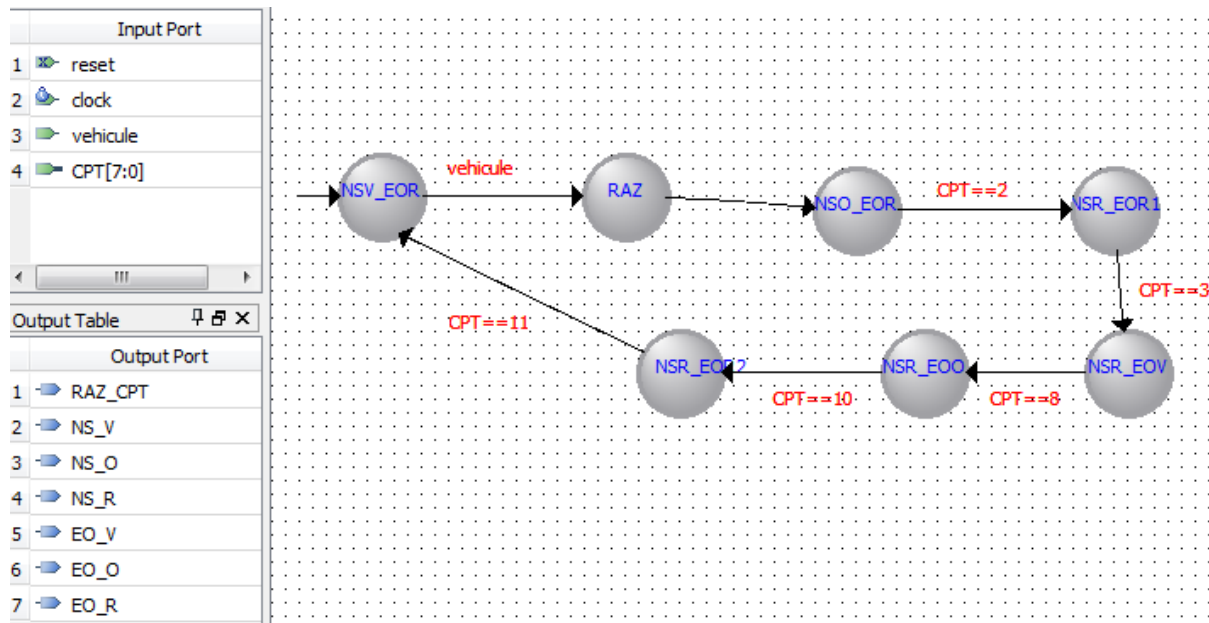
Un détecteur de véhicules est placé sur la voie Est-Ouest (EO). Lorsqu'un véhicule se présente, la voie NS passe à l'orange pour 2 secondes puis au rouge. Après 1 seconde, la voie EO passe au vert pour 5 secondes, puis à l'orange pour 2 secondes, puis au rouge. Après 1 seconde, la voie NS passe au vert et le cycle peut recommencer si un autre véhicule est détecté sur la voie EO.



Modélisation de la machine d'état et génération du symbole correspondant

Modéliser le contrôleur de ce carrefour en utilisant la machine d'états suivante :

TP3 SYS2044




Comme le montre la machine à états, initialement (état NSV_EOR) la voie NS est au vert et la voie EO au rouge. Dès qu'on détecte un véhicule sur la voie EO (entrée « vehicule », on passe à l'état RAZ qui permet de mettre à zéro un compteur qui sert à compter les durées des feux sur les voies. Le compteur donne en sortie la variable CPT qui indique le nombre de cycle de comptage. Cette variable est utilisée par la machine à états pour calculer les durées des feux.

Les sorties de la machine sont RAZ_CPT qui sert à mettre à zéro le compteur, les 6 sorties restantes représentent les 3 feux pour les deux voies. A l'état initial (NSV_EOR), la voie NS est en vert alors que la voie EO est en rouge. Le compteur est mis à zéro dans l'état RAZ.

Créer cette machine à états. Pour aller plus vite par rapport aux valeurs des sorties associées à chaque état, n'ajouter que les cas où la sortie est égale à 1 puisque dans le code VHDL généré, les sorties sont initialisées à 0 à chaque changement des entrées ou/et de l'état.

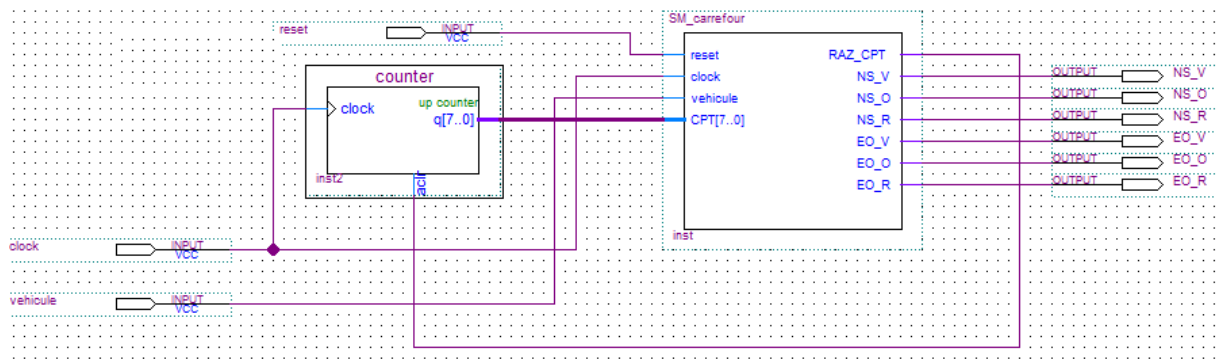
Générer le fichier VHDL correspondant à cette machine d'états, puis le symbole à partir de ce fichier.

Création des composants en entrée à la machine d'états et du schéma global

Ajouter un compteur (Cliquer sur , sélectionner lpm_counter dans megafonctions → arithmetic) que vous nommez counter (IP Catalog à droite, sélectionner library → Basic Functions → Arithmetic → lpm_counter). Ce compteur doit contenir 8 bits de sortie et une entrée de mise à zéro (Clear Asynchrone : aclr).

Construire un schéma qui contient le symbole généré pour la machine d'états et le compteur counter. Faire les connexions comme le montre le schéma ci-dessous.

TP3 SYS2044



Générer un fichier VHDL pour ce schéma.

Mettre ce VHDL en Top Level, désactiver l'optimisation du placement et compiler.

Simulation RTL

Préparer un script de simulation qui affecte les valeurs à l'horloge clock, au reset et à l'entrée vehicule de la manière suivante :

- Clock est un signal d'horloge d'une période de 1s. Le signal a initialement la valeur 0. Il bascule donc à 1 à l'instant 500ms pour repasser à 0 à l'instant 1s ainsi de suite.
- Le signal reset sert à la réinitialisation de la machine à états. Il doit donc commencer avec une valeur de 1 et au bout d'un moment se mettre à 0 et maintenir cette valeur pour le reste de la simulation. La durée de l'impulsion reset n'est pas très importante (exemple : 500 ms).
- Le signal vehicule : ce signal est l'entrée de la machine d'états et puisque l'état de la machine se met à jour sur front montant de l'horloge, il faut que les impulsions de ce signal soit assez large pour coïncider avec ce front montant. Ainsi les impulsions du signal d'horloge doivent être plus larges que 1s pour être sûr sur ça coïncide au moins avec un front montant.

Vous pouvez utiliser Clock et/ou Force pour l'affectation des signaux (clic-droit sur le signal dans la fenêtre de simulation)

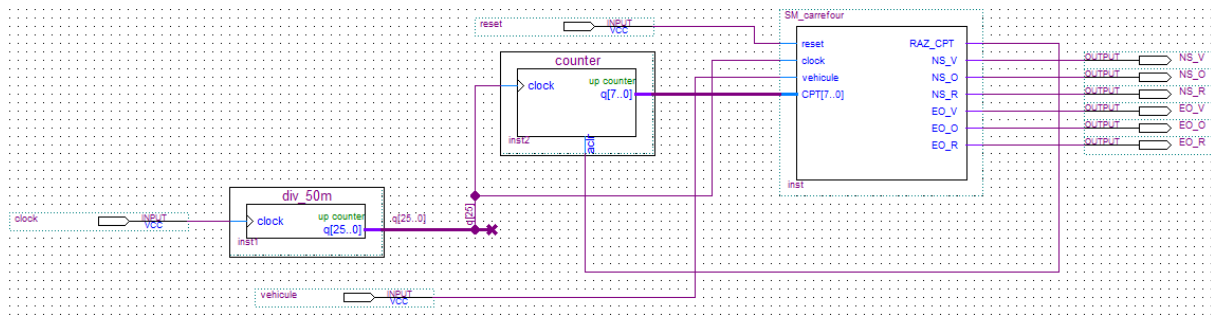
Lancer le simulateur et exécuter le script de simulation. Vérifier que le résultat correspond au cahier de charges.

Appeler l'encadrant pour valider la simulation.

Implémentation physique

Créer un nouveau Schéma. Y copier le contenu du schéma précédant et y ajouter un diviseur de fréquence (diviseur par 50 million en utilisant la même méthode de l'exercice précédent). Le schéma final doit être comme suivi :

TP3 SYS2044



Mettre ce nouveau schéma en Top Level et compiler.

-Faire l'affectation des pins comme suit :

Reset → SW[0]

L'horloge du diviseur de fréquence → PIN_AF14

Vehicule → SW[1]

NS_V → LEDR[9]

NS_O → LEDR[8]

NS_R → LEDR[7]

EO_V → LEDR[2]

EO_O → LEDR[1]

EO_R → LEDR[0]

-Recompiler et Tester sur la carte.

Exercice 4 : gestionnaire de parking

Modélisation de la machine d'états

Comme le montre la figure ci-dessous, la machine d'états qui modélise le gestionnaire du parking a en entrée deux variables E et S qui proviennent de capteurs qui détectent respectivement l'entrée et la sortie d'un véhicule du parking. La machine d'états a également une entrée CPT qui provient d'un compteur/décompteur qui compte **le nombre de places libres**. La machine d'états a en sortie UD envoyée vers le compteur. UD prend la valeur 1 si on veut activer le mode comptage du compteur/décompteur et la valeur 0 si on veut activer le mode décomptage. La sortie CLK_CPT de la machine d'états permet d'activer/stopper le compteur/décompteur, donc pour incrémenter ou décrémenter le compteur/décompteur de 1, il faut mettre CLK_CPT à 1 pour un cycle d'horloge puis le remettre à 0 (le comptage/décomptage se fait sur front montant). Le rôle des sorties UD et CLK_CPT de la machine d'états est d'incrémenter/décrémenter le compteur/décompteur qui indique le nombre de places libres.

TP3 SYS2044

Input Table	
Input Port	
1	reset
2	clock
3	E
4	S
5	CPT[3:0]
Output Table	
Output Port	
1	UD
2	CLK_CPT

E : détection d'une entrée d'un véhicule

S : détection d'une sortie d'un véhicule

CPT : nombre de places disponibles

UD : Up/Down= 1 pour activer le comptage, 0 pour activer le décomptage

CLK_CPT : activation de l'horloge du compteur.

Sachant que **la capacité du parking est de 10 véhicules**, créer la machine d'états du gestionnaire du parking en utilisant les entrées/ sorties décrites ci-dessus.

Les opérateurs utilisables sur les transitions d'une machine à états sont :

TP3 SYS2044

Verilog Operator	Name	Functional Group
[]	bit-select or part-select	
()	parenthesis	
!	logical negation	logical
~	negation	bit-wise
&	reduction AND	reduction
	reduction OR	reduction
~&	reduction NAND	reduction
~	reduction NOR	reduction
^	reduction XOR	reduction
~^ or ^~	reduction XNOR	reduction
+	unary (sign) plus	arithmetic
-	unary (sign) minus	arithmetic
{ }	concatenation	concatenation
{ { } }	replication	replication
*	multiply	arithmetic
/	divide	arithmetic
%	modulus	arithmetic
+	binary plus	arithmetic
-	binary minus	arithmetic
<<	shift left	shift
>>	shift right	shift
>	greater than	relational
>=	greater than or equal to	relational
<	less than	relational
<=	less than or equal to	relational
==	case equality	equality
!=	case inequality	equality
&	bit-wise AND	bit-wise
^	bit-wise XOR	bit-wise
	bit-wise OR	bit-wise
&&	logical AND	logical
	logical OR	logical
?:	conditional	conditional

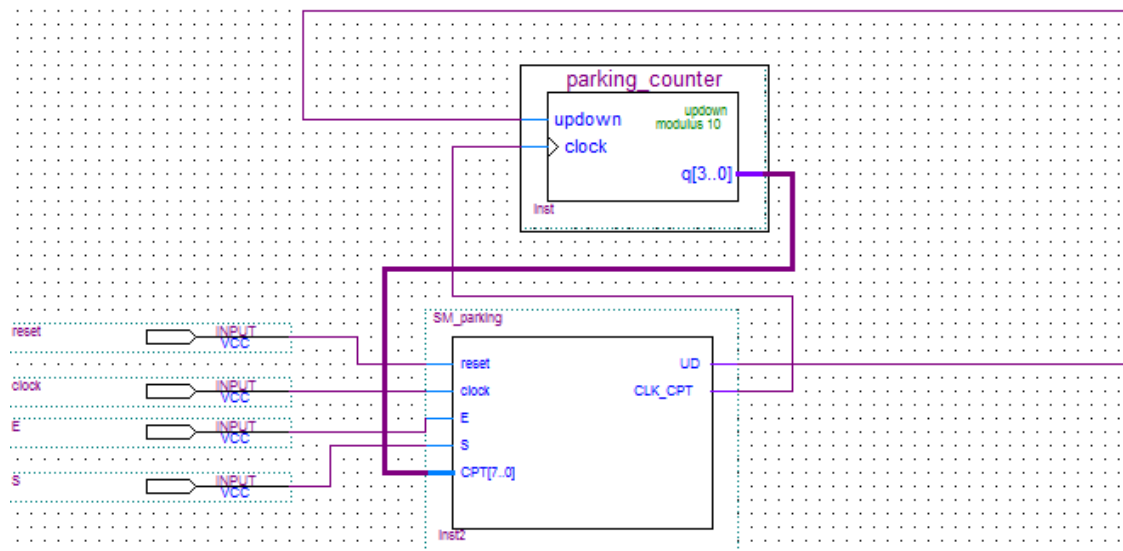
Générez le fichier VHDL correspondant à la machine d'états, désactiver l'optimisation du placement et compiler.

Simulation de la machine d'états

Générer un symbole à partir du fichier VHDL de la machine à états.

Créer un schéma qui contient le symbole généré ainsi qu'un compteur ayant 4 bits de sortie (puisque le nombre maximal de places libres est de 10). Le compteur doit contenir une entrée de comptage/ décomptage. Faire les connexions comme le montre le schéma suivant.

TP3 SYS2044



Générer un fichier VHDL à partir de ce schéma. L'outil génère des signaux correspondant aux connexions entre le compteur et la machine à états. Ces noms sont à retenir pour vérifier leurs valeurs lors de la simulation. Par exemple, le fichier généré contient :

```
b2v_inst : parking_counter2
PORT MAP(updown => SYNTHESIZED_WIRE_0,
         clock => SYNTHESIZED_WIRE_1,
         q => SYNTHESIZED_WIRE_2);
```

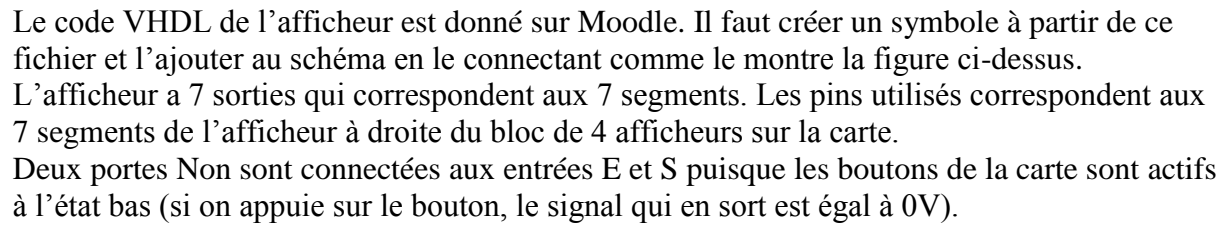
Cela signifie que le signal correspondant à la connexion entre la sortie du compteur et l'entrée CPT de la machine d'états s'appelle SYNTHESIZED_WIRE_2. L'entrée updown du compteur et la sortie UD de la machine d'états seront connectés par le signal SYNTHESIZED_WIRE_0, etc.

Mettre le fichier VHDL en Top-Level, compiler et lancer une simulation pour vérifier le bon fonctionnement du système : le compteur est incrémenté si un véhicule sort du parking et décrémenter si un véhicule y entre. Il faut vérifier également que la valeur du compteur ne dépasse pas 10.

Implémentation sur la carte

Pour tester ce système sur la carte, les entrées/sorties des véhicules seront simulées par un appui sur deux boutons différents. Le nombre de places libres sera affiché sur un afficheur 7-segments.

Le schéma final est le suivant :



20