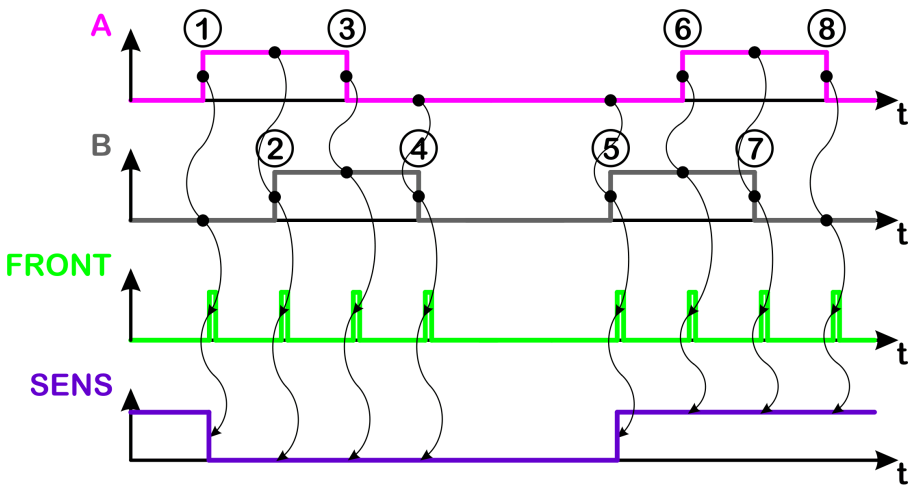
BTS SN	FORMATION ALTIUM CODEUR INCRÉMENTAL VHDL + FSM + SIMULATION	TP10 -4-

Altium Designer



I. PRÉSENTATION DE LA FONCTION FP4

Cette fonction est chargée de détecter chaque front des signaux A et B issus du codeur incrémental et par la même d'en déterminer le sens de rotation.

I.1. DESCRIPTION DU COMPOSANT CODEUR_INCRÉMENTAL

Une entrée de RESET synchrone permettra de placer les sorties dans un état lors de la mise sous tension déterminé (initialisation de la logique séquentielle).

Suivant l'évolution des signaux A et B qui sont déphasés de 90°, le signal VALIDATION passe à 1 pendant 1 coup d'horloge à chaque changement de A ou B.
Si le signal A est en avance sur le signal B alors SENS passe à 1, sinon il prend la valeur 0.

La synthèse peut être faite de plusieurs manières différentes : soit l'écriture directe d'un fichier VHDL soit le dessin d'une machine à états (FSM ou Finite State Machine) ou encore le dessin d'une structure électronique (schéma). Nous en verrons les deux premières solutions.

I.2. CHRONOGRAMMES ATTENDUS

A l'analyse des signaux A et B on remarque qu'il y a 8 cas différents : à chaque front d'un signal, l'autre peut être à 0 ou à 1.
La sortie FRONT passera à 1 fugitivement (durée un coup d'horloge) à chaque fois qu'un front sur A ou B se présentera.
La sortie SENS Passera à 0 pour les cas ① à ④ et à 1 pour les cas ⑤ à ⑧.

II. ÉDITION D'UN FICHIER VHDL

II.1. AJOUT AU PROJET D'UN NOUVEAU FICHIER VHDL

Ajouter un nouveau fichier VHDL au projet. Renommer-le **Codeur_Incremental.Vhd**.

II.2. ÉDITION ET COMPILATION DU FICHIER VHDL

Éditer le fichier VHDL, enregistrer-le puis compiler-le, Corriger les erreurs qui pourraient apparaître.
(Voir listing en annexe pages 9 et 10).

Remarque : le principe est d'utiliser des registres à décalages (bascules D) afin de détecter si un front vient d'avoir lieu sur l'entrée A ou B. Dans ce cas la valeur actuelle est différente de la valeur précédente. Il ne reste plus qu'à lister les différents cas.

III. SIMULATION FONCTIONNELLE AVEC ALDEC OEM

III.1. PRODUCTION DU FICHIER DE TEST

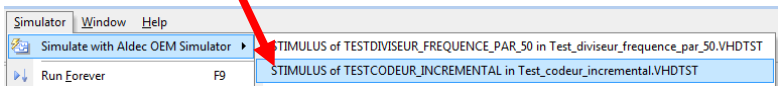
Créer le fichier VHDL de test.
Ajouter à celui-ci les lignes des process de tests afin de faire évoluer les ports d'entrées suivant un jeu d'essai permettant de vérifier tous les cas. (Voir listing en annexe page 11).
Enregistrer celui-ci en gardant le nom proposé.
Compiler ce fichier puis compiler le projet complet.

III.2. ASSOCIATION DES FICHIERS

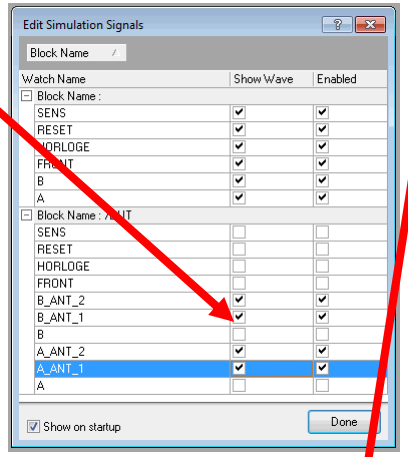
Ajouter le fichier de test dans le Manager Testbenches du simulateur.

III.3. SIMULATION

Lancer la simulation :

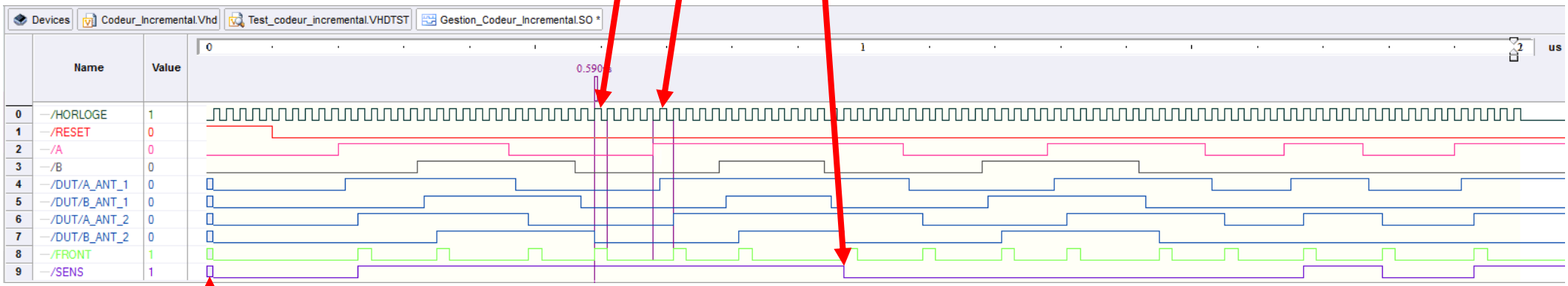


Cocher les signaux à visualiser (show Wave) en plus des ports d'entrées/sorties : (A_ANT_1, B_ANT_1, A_ANT_2, B_ANT_2).



Lancer la simulation sur une durée de 2 µs.

Afficher la simulation complète.
Replacer les différents signaux dans un ordre logique.



On peut remarquer que les signaux internes (DUT) et les sorties ne sont pas définies avant le premier front montant de l'horloge HORLOGE qui les place dans l'état défini dans le fichier VHDL puisque l'entrée RESET est synchrone.

Analyser les chronogrammes obtenus.

On peut remarquer que le signal FRONT est activé pendant un coup d'horloge à chaque fois qu'un front sur A ou B survient.

Le signal FRONT passe au niveau haut au maximum 2 coups d'horloge après le changement d'un des deux signaux d'entrée A ou B.

Le signal SENS respecte bien l'évolution prévue.

Sauvegarder la fenêtre de simulation.
Sauvegarder le projet complet.

La fonction FP4 est donc validée.

Il peut parfois être utile de proposer d'autres excitations afin d'observer le comportement du programme VHDL, en particulier des cas (ou jeux d'essais) qui normalement ne se produisent pas en fonctionnement normal mais peuvent apparaître dans la réalité (rebonds, etc).

Effectuer une nouvelle simulation en modifiant le fichier de TestBench comme indiqué pour voir l'incidence de rebonds lors du changement d'état de l'entrée A (Voir listing partiel en annexe page 12).

On peut remarquer que les rebonds haute fréquence sont pris en compte par le composant. Le compteur qui suit comptera 7 coups au lieu d'un seul.
Il faut donc prévoir un composant qui supprimera les rebonds en amont (FP1 et FP2).

IV. ÉDITION D'UNE MACHINE À ÉTATS

IV.1. PRÉSENTATION RAPIDE

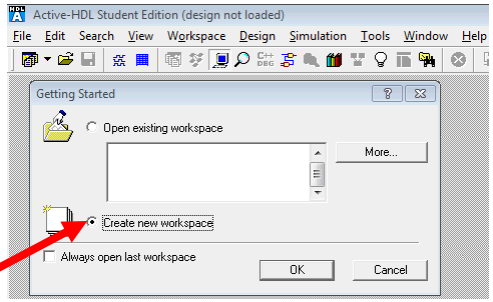
Une machine à états (ou FSM en anglais) est la représentation graphique d'un structure électronique très utilisée en logique séquentielle câblée (séquenceurs utilisant souvent des bascules JK).
Elle fait appel à des états (ou étapes) et des transitions.

Altium ne dispose pas d'éditeur de machines à états. Nous allons donc utiliser celui proposé par ALDEC (le second grand de la synthèse logique).
Il faut pour cela installer **ALDEC Active-HDL Student Edition**.
Cette version limitée est gratuite pour les étudiants et les organismes de formation.

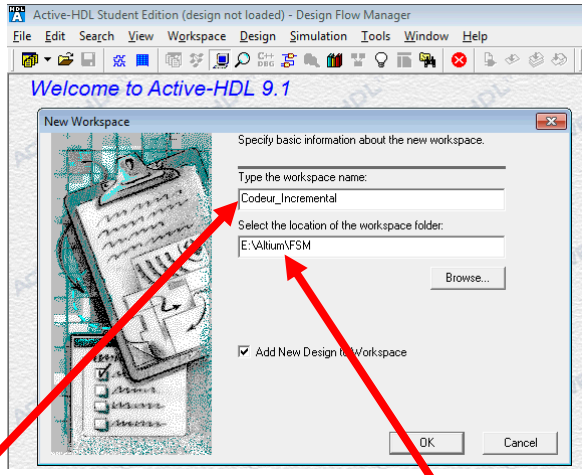
Dans cette partie nous considérerons que le logiciel est correctement installé.

IV.2. CRÉATION D'UN ESPACE DE TRAVAIL

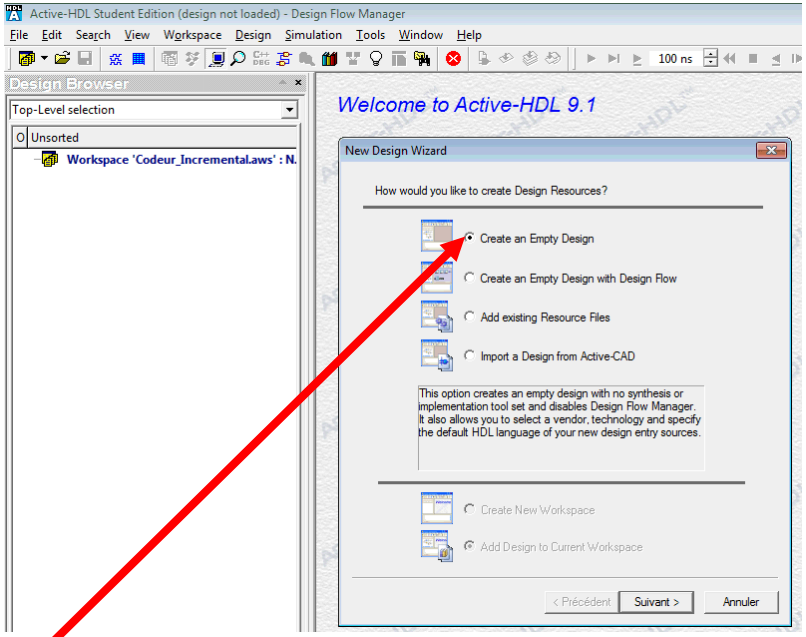
Lancer le logiciel **ALDEC Active-HDL Student Edition** en double-cliquant sur son icone.



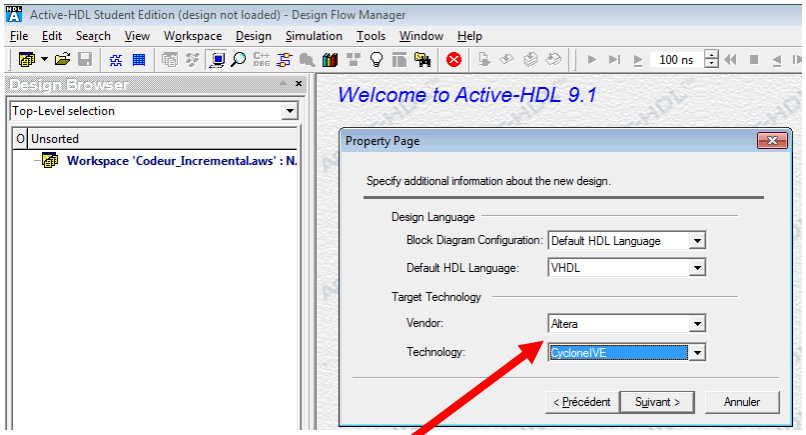
Sélectionner **Create new workspace** puis OK.



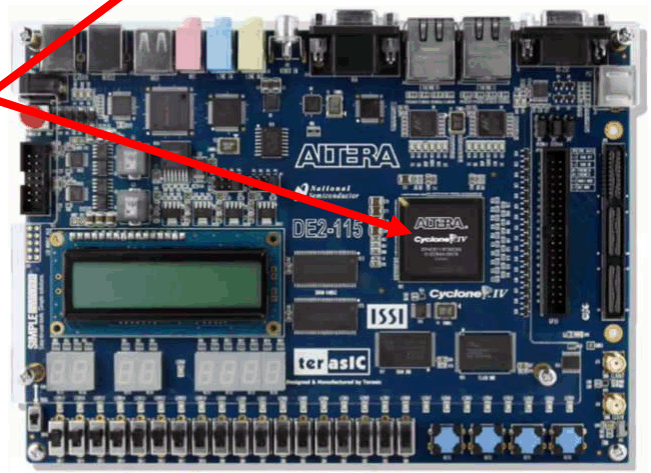
Donner un nom à votre espace de travail (projet).
Préciser le chemin à partir duquel un répertoire de travail sera créé (différent de celui du projet Altium). Puis OK.



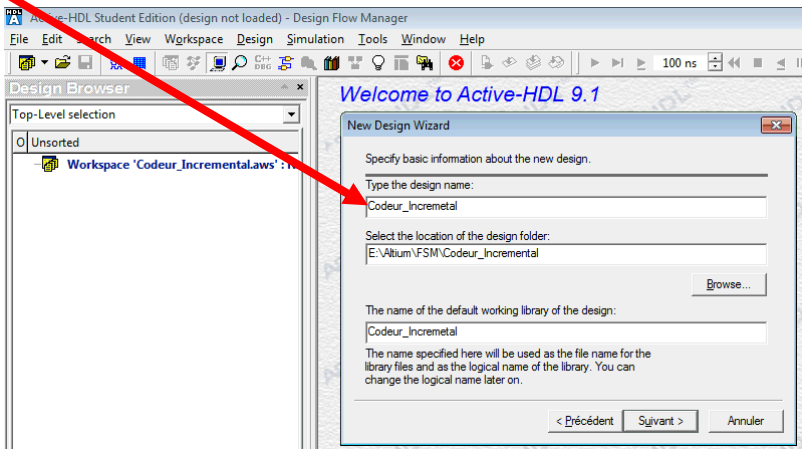
Sélectionner **Create an Empty Design** puis Suivant.

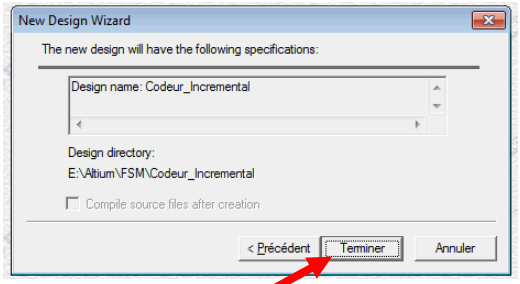


Renseigner, si besoin, ces deux menus déroulants en fonction de la cible utilisée. Ici ce sera un Cyclone IV d'Altera car la carte utilisée est une DE2-115 de TERASIC. Puis faire suivant.



Nommer le Design Codeur_Incremental puis faire Suivant.

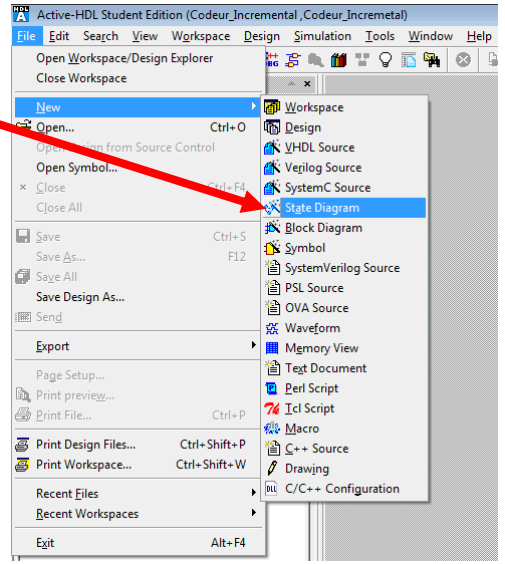




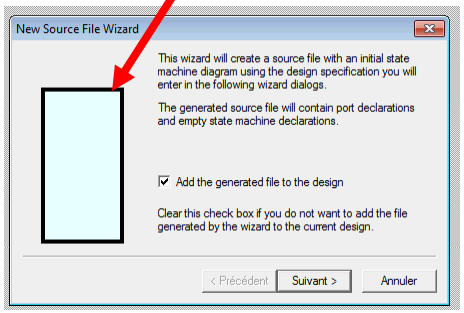
Sur la dernière fenêtre cliquer sur Terminer.

IV.3. CRÉATION DE LA MACHINE À ÉTATS

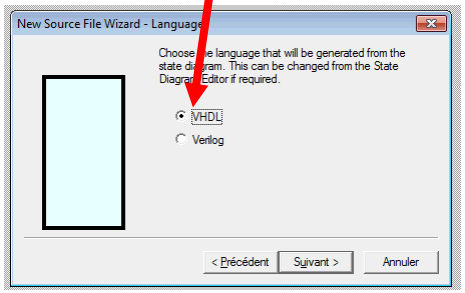
Créer une nouvelle machine à états :
⇒ **File ⇒ State Diagram**



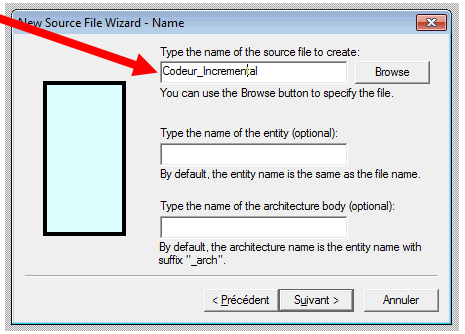
Un wizard (assistant) va permettre de construire simplement l'entité. Faire Suivant.



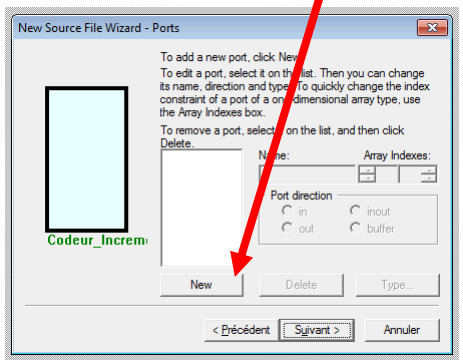
Langage utilisé : VHDL.



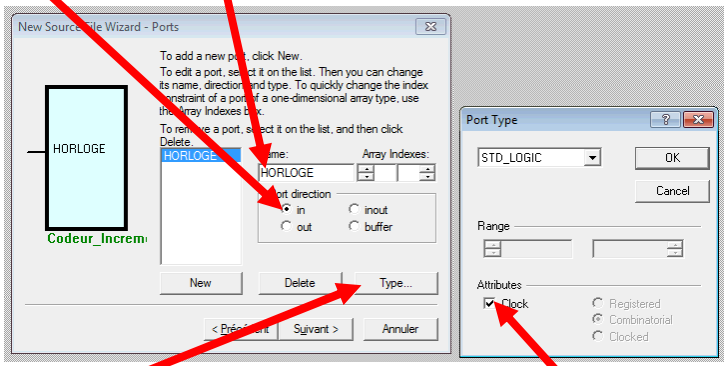
Nom de la machine à états : Codeur_Incremental.



Cette fenêtre permet d'ajouter des ports à l'entité. Faire New.



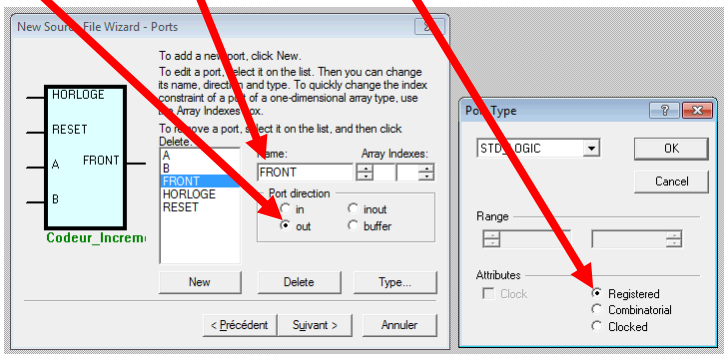
Ajouter en entrée le port HORLOGE.



Puis cliquer sur Type pour faire apparaître la fenêtre permettant d'indiquer que c'est un signal d'horloge. Faire OK puis New.

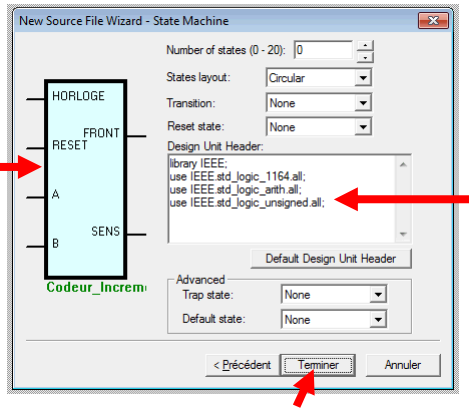
Ajouter les 3 autres entrées : RESET, A et B (par défaut les entrées ne sont pas des signaux d'horloges).

Ajouter en sortie le port FRONT de type Registered (machine de Moore). Faire OK puis New.



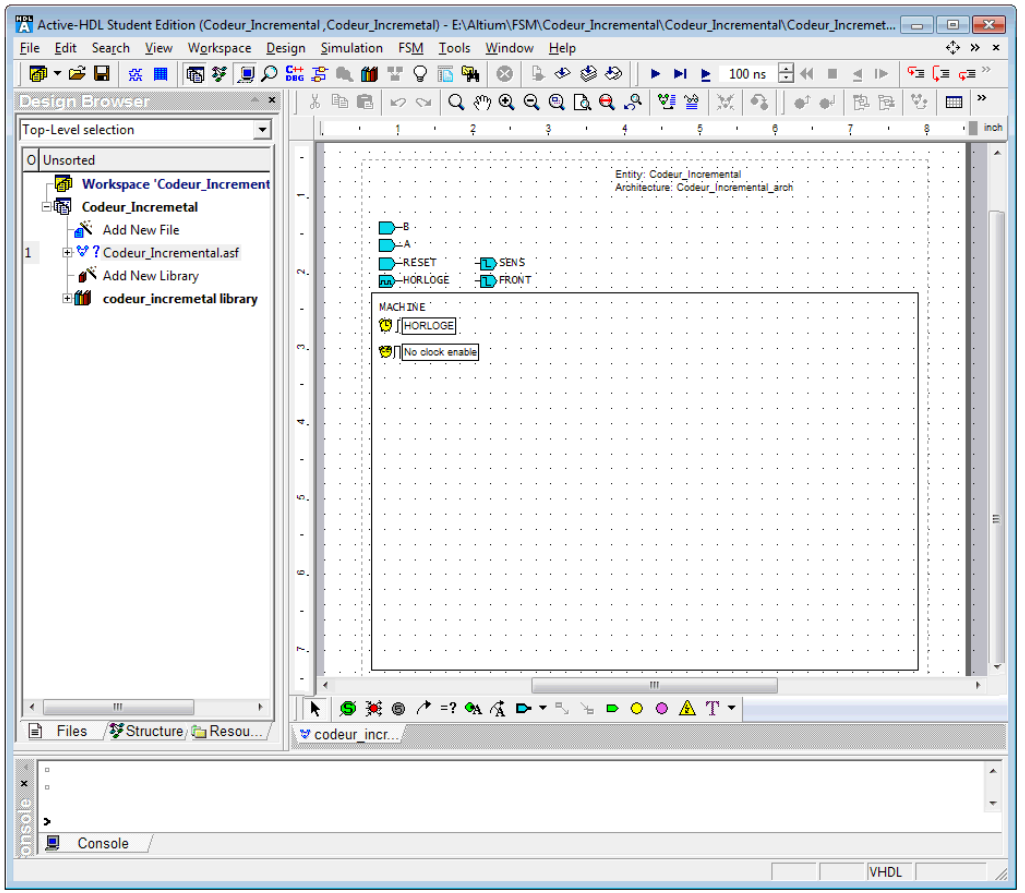
Ajouter l'autre sortie : SENS du même type puis faire Suivant.

Entité créée



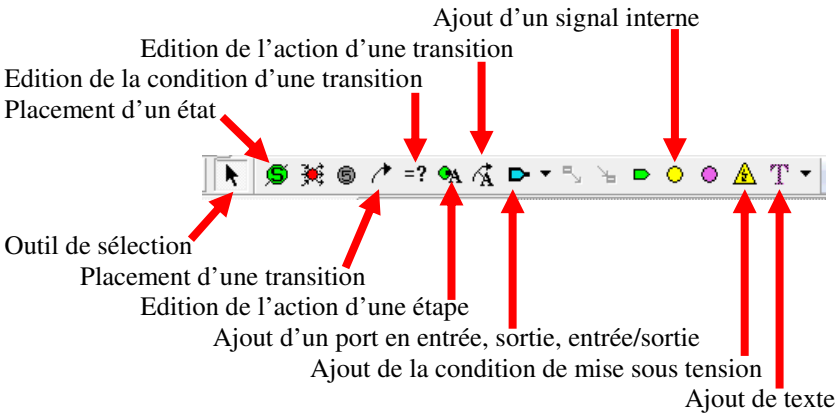
Librairies utilisées

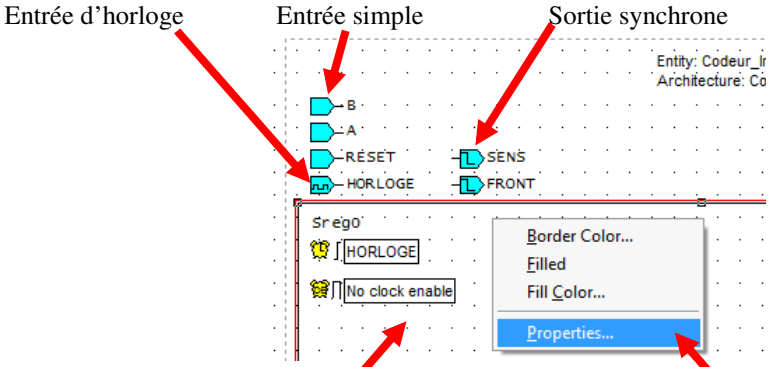
Il est possible de définir d'autres paramètres ici mais ce sera fait manuellement plus tard. Faire terminer.



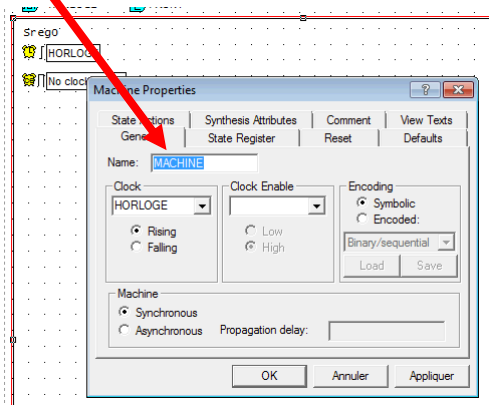
IV.4. DESSIN DE LA MACHINE À ÉTATS

Barre d'outils :



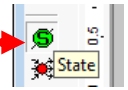


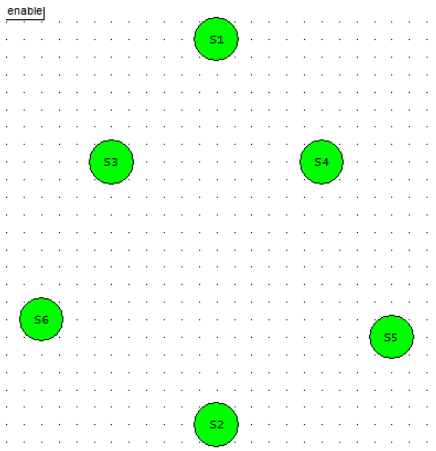
Cliquer droit dans la fenêtre de la machine à états puis cliquer sur Properties… et renseigner les propriétés comme ci-dessous :



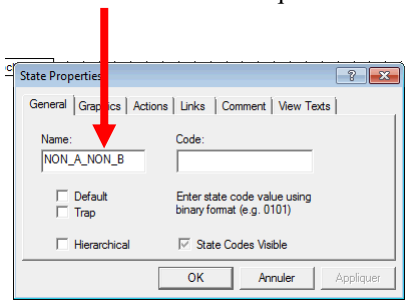
La machine à états complète est donnée en annexe page 12 pour aider au dessin.

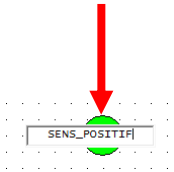
Disposer 6 états comme ci-dessous avec cet outil :



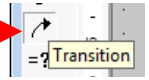


Renommer les états soit en double cliquant dessus soit en cliquant sur le nom comme donné ci-dessous.

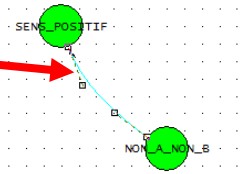




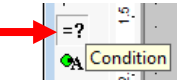
Ajouter les transitions entre les états avec cet outil :

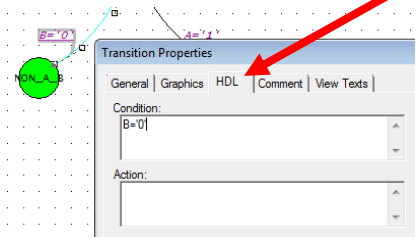


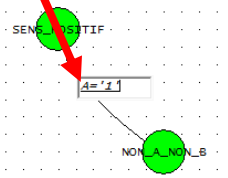
Cliquer sur le bord de l'état de départ puis sur le bord de l'état d'arrivée. Une flèche apparait. En cliquant dessus on peut faire apparaître deux points qui permettent de courber la transition.
Il est aussi possible de déplacer les points de départ et d'arrivée en cliquant dessus.



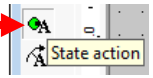
Éditer la condition de la transition en cliquant dessus ou avec cet outil :

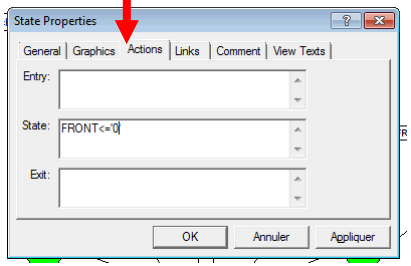




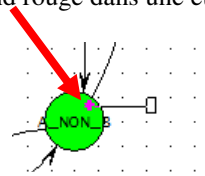


Ajouter les actions soit en cliquant sur les états soit avec cet outil :

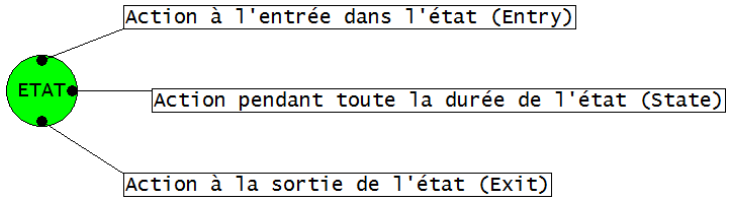




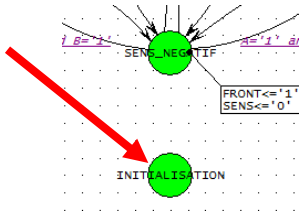
Placer le rond rouge dans une étape



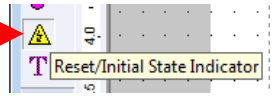
Suivant le besoin 3 types d'actions peuvent être choisis :

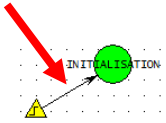


Ajouter un dernier état :



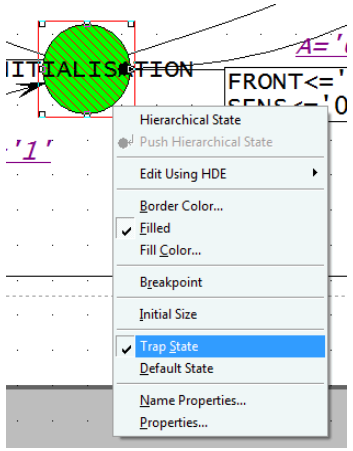
Ajouter une condition de mise sous tension avec cet outil :





Puis éditer la condition correspondant à cette transition.

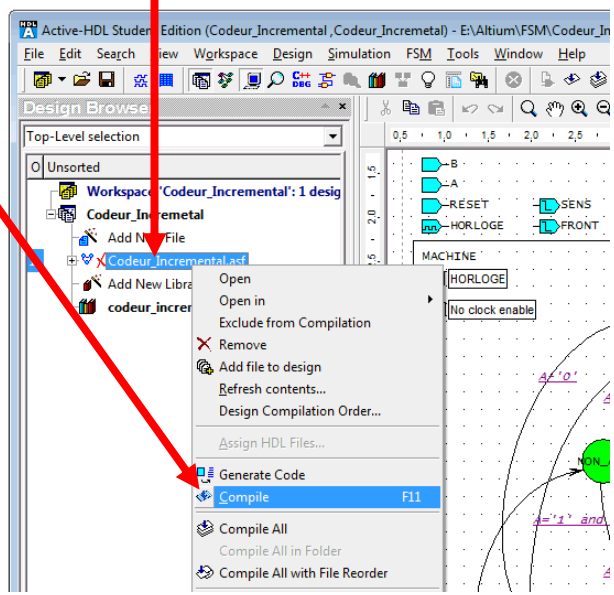
Il ne reste plus qu'à définir un des états comme l'état piège, c'est-à-dire que s'il arrive une erreur au niveau de la machine à états c'est dans cet état que la machine retournera.
Dans notre cas il y a 7 états, il faut donc 3 bits au minimum pour les coder. Il reste donc un état inutilisé, si par malheur la machine entrait dedans elle pourrait y rester indéfiniment. Le fait de définir un piège permet d'en sortir le coup d'horloge suivant.
Cliquer droit sur l'état et cocher Trap State.



La machine à états est maintenant complète. Enregistrer celle-ci.
A remarquer qu'il n'y a pas besoin d'une grande connaissance de la syntaxe VHDL à par l'écriture et la syntaxe des conditions et des actions.

IV.5. TRADUCTION DU GRAPHIQUE EN VHDL

Dans la fenêtre des fichiers cliquer droit sur le nom de la machine à états et cliquer sur Compiler.



Dans la fenêtre Console on peut connaitre le résultat de la compilation :

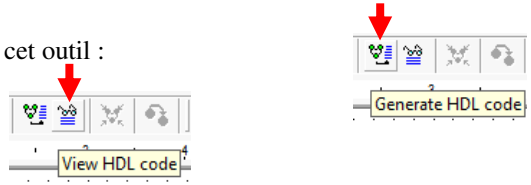
```
# Warning: FSM2HDL_3127 Codeur_Incremental.asf : State INITIALISATION was selected as the trap state. The coverage
pragmas have been omitted.
# HDL code generation completed. 1 error(s), 2 warning(s)
# Double-click on error/warning line to see the source of error/warning.
```

S'il y a des erreurs, les corriger jusqu'à obtenir :

```
# File: .\Codeur_Incremental.vhd
# Compile Entity "Codeur_Incremental"
# Compile Architecture "Codeur_Incremental_arch" of Entity "Codeur_Incremental"
# Compile success 0 Errors 0 Warnings  Analysis time :  0.1 [s]
```

Générer le fichier VHDL à partir de la machine à états avec cet outil :

Afficher le fichier VHDL avec cet outil :



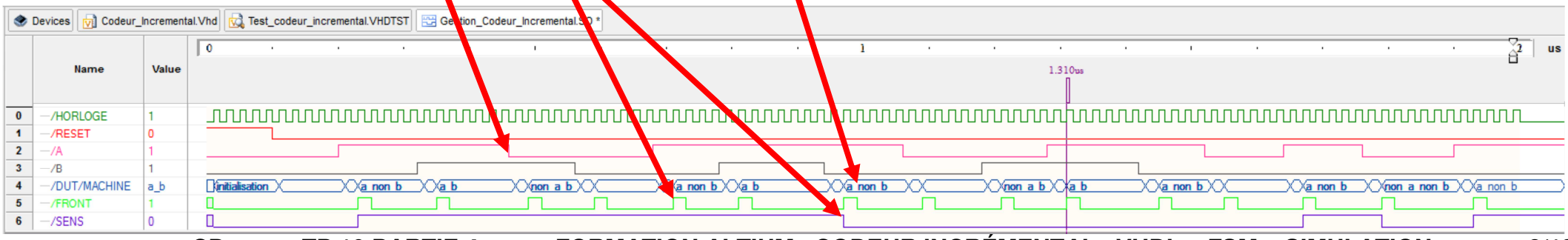
Effectuer un copier de tout le fichier généré (CTRL + A, CTRL + C) et coller le fichier dans Codeur_Incremental.vhd du projet Altium à la place du listing édité précédemment.

Compiler celui-ci, il ne devrait pas y avoir d'erreurs.

V. SIMULATION FONCTIONNELLE AVEC ALDEC OEM

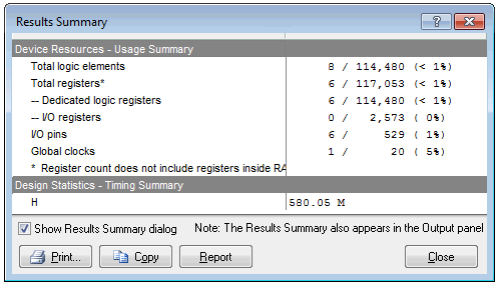
Le nouveau fichier possède les mêmes entrées et sorties et le même fonctionnement que le précédent, il est donc possible de le simuler avec le même fichier de tests.
Seuls les signaux internes sont différents.

On peut remarquer que les états de la machine à états apparaissent bien.
Cela peut aider au débogage.
Les sorties FRONT et SENS sont identiques à la première simulation. Les deux solutions, bien que très différentes donnent le même résultat.
Le retard est ici aussi de 2 coups d'horloges au maximum.

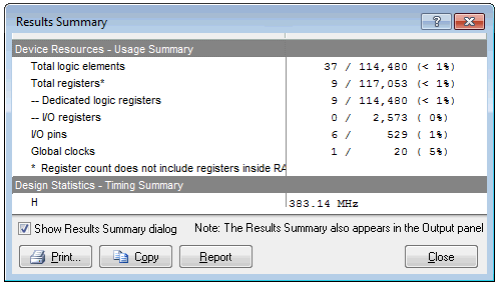


Le meilleur choix pourrait alors être celui mettant en œuvre le minimum de ressources et/ ou le plus rapide. Seul le codeur incrémental a été placé (fitter) dans le FPGA, voici les résultats :

Première solution :



Seconde solution :



On peut remarquer que la solution avec machine à états est plus gourmande en ressources : 37 éléments logiques au lieu de 8, 9 registres (bascules) au lieu de 6.
La vitesse maximum de l'horloge pour la première solution est de 580 MHz alors que pour la seconde est n'est que de 383 MHz. Dans notre cas ce n'est pas critique.

La fonction FP4 est à nouveau validée.

VI. CRÉATION D'UN SYMBOLE POUR LA FONCTION FP3

Vérifier que la fenêtre de travail est bien le fichier VHDL **Codeur_Incremental.Vhd** puis créer un symbole avec la commande :
⇒ **Design** ⇒ **Create Shematic Part From File**

Faire **Ok**. Le symbole apparait alors à l'écran.

Sauvegarder le sous le nom **Codeur_Incremental.SchLib**
Fermez-le : il nous servira plus tard.

VII. ANNEXES

```
1   -- Projet Gestion_Codeur_Incremental
2   -- CODEUR_INCREMENTAL
3
4   -- Description VHDL d'un composant qui détecte les fronts des entrées A et B
5   -- et le sens de rotation d'un codeur incrémental
6   -- sortie FRONT indiquant qu'un déplacement vient d'avoir lieu (actif à 1)
7   -- sortie SENS indiquant le sens du déplacement qui vient d'avoir lieu
8   -- 08/11/2012  C. D. Lycée Chevrollier Angers
9
10  library IEEE;                                   --Liste des librairies prédéfinies utilisées
11  use IEEE.STD_LOGIC_1164.ALL;
12  use IEEE.STD_LOGIC_ARITH.ALL;
13  use IEEE.STD_LOGIC_UNSIGNED.ALL;
14
15  entity CODEUR_INCREMENTAL is                    --Description de l'entité (broches du composant)
16    port (
17          HORLOGE           : in  std_logic;      --Entrée de synchronisation
18          RESET             : in  std_logic;      --Entrée d'initialisation (systèmes synchrone)
19          A                 : in  std_logic;      --Entrée A connectée au codeur incrémental
20          B                 : in  std_logic;      --Entrée B connectée au codeur incrémental
21          FRONT             : out std_logic;      --Sortie indiquant qu'un déplacement à eu lieu
22          SENS              : out std_logic       --Sortie indiquant le sens du déplacement
23        );
24  end CODEUR_INCREMENTAL;                         --Fin de la description de l'entité
25
26  architecture ARCH_CODEUR_INCREMENTAL of         --Description de l'architecture
27               CODEUR_INCREMENTAL is              --Comportement interne du composant (silicium)
28    signal  A_ANT_1     : std_logic;              --Valeur du signal A il y a 1 coup d'horloge
29    signal  B_ANT_1     : std_logic;              --Valeur du signal B il y a 1 coup d'horloge
30    signal  A_ANT_2     : std_logic;              --Valeur du signal A il y a 2 coups d'horloge
31    signal  B_ANT_2     : std_logic;              --Valeur du signal B il y a 2 coups d'horloge
32    begin                                         --Début de la description synchrone
33      process (HORLOGE)                           --Partie synchrone sur le signal HORLOGE
34        variable VALEURS : std_logic_vector (3 downto 0); -- Variable servant à la concaténation
35        begin                                     --Fonctionnement complètement synchrone
36          VALEURS := A_ANT_2 & B_ANT_2 &          --Déclaration d'une variable pour concaténer
37                     A_ANT_1 & B_ANT_1;           --les 4 bits pour détection de front
38          if (HORLOGE'event and HORLOGE = '1')    --Attente d'un font montant du signal HORLOGE
39            then
40              if (RESET = '1')                    --Si RESET actif (à la mise sous tension)
41                then
42                  A_ANT_1 <= A;                   --Initialise les signaux internes
43                  A_ANT_2 <= A;                   --Initialise les signaux internes
44                  B_ANT_1 <= B;                   --Initialise les signaux internes
45                  B_ANT_2 <= B;                   --Initialise les signaux internes
46                  FRONT <= '0';                   --Initialise les ports en sorties
47                  SENS <= '0';                    --Initialise les ports en sorties
```

```
48              else                            --Sinon
49                 A_ANT_1 <= A;                --Mémorisation de la nouvelle valeur de A
50                 B_ANT_1 <= B;                --Mémorisation de la nouvelle valeur de B
51                 A_ANT_2 <= A_ANT_1;          --Mémorisation de l'ancienne valeur de A
52                 B_ANT_2 <= B_ANT_1;          --Mémorisation de l'ancienne valeur de B
53                 case VALEURS is              --Début du choix multiple
54                   when "0010" =>             --Cas n°1
55                     FRONT <= '1';            --Informe qu'un font a été détecté
56                     SENS <= '1';             --Informe du sens positif
57                   when "1011" =>             --Cas n°2
58                     FRONT <= '1';            --Informe qu'un font a été détecté
59                     SENS <= '1';             --Informe du sens positif
60                   when "1101" =>             --Cas n°3
61                     FRONT <= '1';            --Informe qu'un font a été détecté
62                     SENS <= '1';             --Informe du sens positif
63                   when "0100" =>             --Cas n°4
64                     FRONT <= '1';            --Informe qu'un font a été détecté
65                     SENS <= '1';             --Informe du sens positif
66                   when "0001" =>             --Cas n°5
67                     FRONT <= '1';            --Informe qu'un font a été détecté
68                     SENS <= '0';             --Informe du sens négatif
69                   when "0111" =>             --Cas n°6
70                     FRONT <= '1';            --Informe qu'un font a été détecté
71                     SENS <= '0';             --Informe du sens négatif
72                   when "1110" =>             --Cas n°7
73                     FRONT <= '1';            --Informe qu'un font a été détecté
74                     SENS <= '0';             --Informe du sens négatif
75                   when "1000" =>             --Cas n°8
76                     FRONT <= '1';            --Informe qu'un font a été détecté
77                     SENS <= '0';             --Informe du sens négatif
78                   when others =>             --Dans tous les autres cas
79                     FRONT <= '0';            --Pas de font détecté
80                 end case;                    --Fin du choix multiple
81              end if;                         --Fin de la condition sur l'entrée RESET
82           end if;                            --Fin de la condition sur l'entrée HORLOGE
83       end process;                           --Fin de la description synchrone
84   end ARCH_CODEUR_INCREMENTAL;               --Fin de la description interne
85
```

```vhdl
--------------------------------------------------------------------
-- VHDL Testbench for codeur_incremental
-- 2012 11 10 8 57 38
-- Created by "EditVHDL"
-- "Copyright (c) 2002 Altium Limited"
--------------------------------------------------------------------

Library IEEE;
Use     IEEE.std_logic_1164.all;
Use     IEEE.std_logic_textio.all;
Use     STD.textio.all;
--------------------------------------------------------------------

--------------------------------------------------------------------
entity Testcodeur_incremental is
end Testcodeur_incremental;
--------------------------------------------------------------------

--------------------------------------------------------------------
architecture stimulus of Testcodeur_incremental is
    file RESULTS: TEXT open WRITE_MODE is "results.txt";
    procedure WRITE_RESULTS(
        A: std_logic;
        B: std_logic;
        FRONT: std_logic;
        HORLOGE: std_logic;
        RESET: std_logic;
        SENS: std_logic
    ) is
        variable l_out : line;
    begin
        write(l_out, now, right, 15);
        write(l_out, A, right, 2);
        write(l_out, B, right, 2);
        write(l_out, FRONT, right, 2);
        write(l_out, HORLOGE, right, 2);
        write(l_out, RESET, right, 2);
        write(l_out, SENS, right, 2);
        writeline(RESULTS, l_out);
    end procedure;

    component codeur_incremental
        port (
            A: in std_logic;
            B: in std_logic;
            FRONT: out std_logic;
            HORLOGE: in std_logic;
            RESET: in std_logic;
            SENS: out std_logic
        );
    end component;

    signal A: std_logic;
    signal B: std_logic;
    signal FRONT: std_logic;
    signal HORLOGE: std_logic;
    signal RESET: std_logic;
    signal SENS: std_logic;

begin
    DUT:codeur_incremental port map (
        A => A,
        B => B,
        FRONT => FRONT,
        HORLOGE => HORLOGE,
        RESET => RESET,
        SENS => SENS
    );

    GENERATION_HORLOGE:process
    begin
        HORLOGE <= '0';
        wait for 10 ns;
        HORLOGE <= '1';
        wait for 10 ns;
    end process;

    STIMULUS0:process
    begin
        -- insert stimulus here
      RESET <= '1';
      A <= '0';
      B <= '0';
      wait for 100 ns;
      RESET <= '0';
      wait for 100 ns;
      A <= '1';
      B <= '0';
      wait for 120 ns;
      A <= '1';
      B <= '1';
      wait for 140 ns;
      A <= '0';
      B <= '1';
      wait for 100 ns;
      A <= '0';
      B <= '0';
      wait for 100 ns;
      A <= '1';
      B <= '1';
      wait for 160 ns;
      A <= '1';
      B <= '0';
      wait for 120 ns;
      A <= '0';
      B <= '0';
      wait for 120 ns;
      A <= '0';
      B <= '1';
      wait for 100 ns;
      A <= '1';
      B <= '1';
      wait for 140 ns;
      A <= '1';
      B <= '0';
      wait for 100 ns;
      A <= '0';
      B <= '0';
      wait for 120 ns;
      A <= '1';
      B <= '0';
      wait for 120 ns;
      A <= '0';
      B <= '0';
      wait for 140 ns;
      A <= '1';
      B <= '0';
      wait for 120 ns;
      A <= '0';
      B <= '0';
      wait;
    end process;

    WRITE_RESULTS(
        A,
        B,
        FRONT,
        HORLOGE,
        RESET,
        SENS
    );

end architecture;
----------------------

----------------------
```

```
70  GENERATION_HORLOGE:process
71  begin
72      HORLOGE <= '0';
73      wait for 10 ns;
74      HORLOGE <= '1';
75      wait for 10 ns;
76  end process;
77
78  STIMULUS0:process
79  begin
80      -- insert stimulus here
81    RESET <= '1';
82    A <= '0';
83    B <= '0';
84    wait for 60 ns;
85    RESET <= '0';
86    wait for 80 ns;
87    A <= '1';
88    B <= '0';
89    wait for 60 ns;
90    A <= '0';
91    B <= '0';
92    wait for 60 ns;
93    A <= '1';
94    B <= '0';
95    wait for 60 ns;
96    A <= '0';
97    B <= '0';
98    wait for 60 ns;
99    A <= '1';
100   B <= '0';
101   wait for 60 ns;
102   A <= '0';
103   B <= '0';
104   wait for 60 ns;
105   A <= '1';
106   B <= '0';
107   wait;
108 end process;
```