Simulation comportementale avec Vivado Simulator (XSIM)

Introduction

Dans le tutoriel précédent (4 - Projet RTL (VHDL) basique) nous avons créé un projet RTL (VHDL) simple. Dans ce tutoriel, nous allons utiliser Vivado Simulator (XSIM) pour valider le comportement de notre design.

Lancer la simulation

Pour lancer le Vivado Simulator en simulation comportementale, cliquez sur "*Run Simulation*" sous "*Simulation*" dans le "*Flow Navigator*" et cliquez ensuite sur "*Run Behavioral Simulation*". Cela ouvrira une "*waveform window*" dans le simulateur.

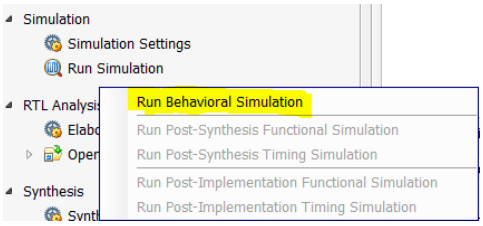

Figure 1 - Run Behavioral Simulation

Dans la "waveform window", sur la gauche, on peut voir les 4 entrées et la sortie de notre design et les signaux intermédiaires que nous avons créé pour les sorties du OU logique et du premier ET logique. La valeur de ces signaux est pour l'instant 'U' (pour Uninitialized) ce qui signifie que nos signaux non pas été initialisés. C'est parce que nous n'avons pas donné de valeur à nos signaux d'entrées.

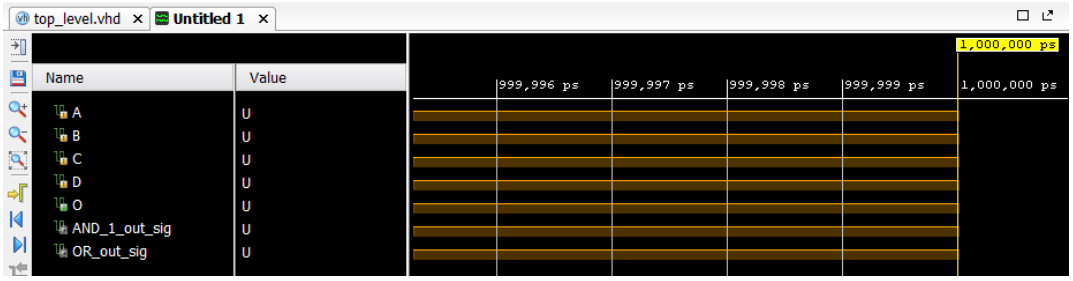

Figure 2 - Waveform window dans Vivado

Nous allons donner une valeur à nos signaux d'entrées en forçant leur valeur dans la simulation. Cliquez sur "*Run > Restart the Simulation* pour redemarrer la simulation.

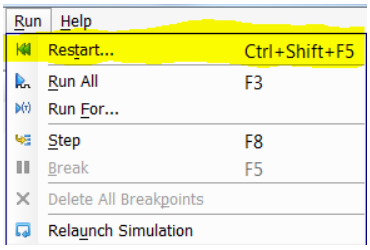

Figure 3 - Restart the simulation - Vivado Simulator

Ensuite nous allons forcer la valeur de nos signaux. Pour tester toutes les possibilités de notre system nous allons forcer ses signaux avec une horloge. Faites un clique droit sur le signal A et cliquez sur "*Force Clock".*

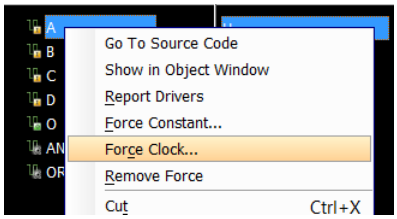

Figure 4 - Force Clock

Changer les valeurs des paramètres "*Value radix"* en "*Binary*", "*Leading edge value"* en '0', "*Trailing edge value*" en '1' et régler la période à "*10ns*".

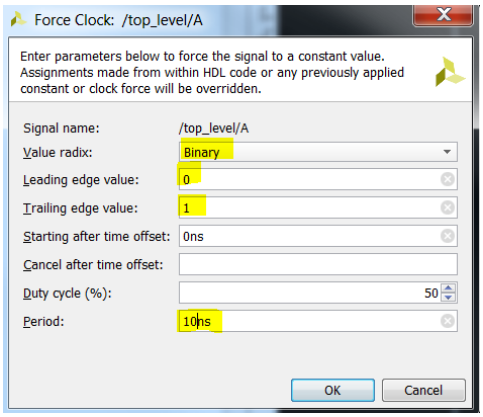

Figure 5 - Force clock sur l'entrée A

Faire de même pour les entrées B, C and D mais avec, respectivement, des périodes de"*20ns*", "*40ns*" et "*80ns*".

Ensuite, lancer la simulation en cliquant sur "*Run > Run For… "*. Régler un temps de simulation de 160ns.

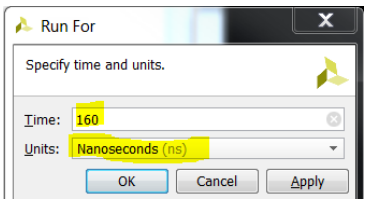

Figure 6 – Lancer la simulation pour 160ns

Dans la "*waveform window*", on peutvoir que la sortie O est à '1' dans 3 cas:

- Quand A, B et C sont à '1' et D est à 0
- Quand A, B et D sont à '1' et C est à 0
- Quand A, B, C et D sont à '1'

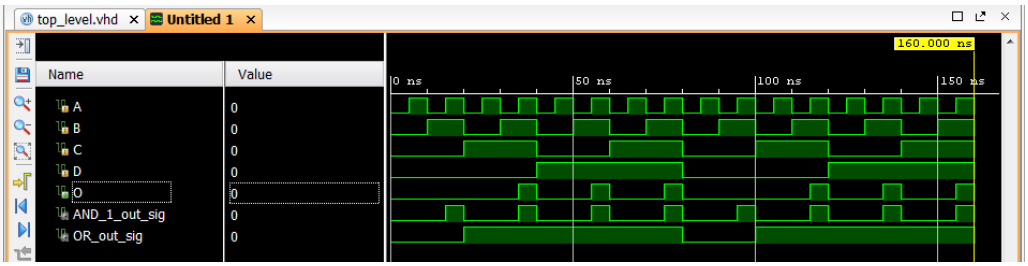

Figure 7 - Vivado Simulator Waveform

C'est le comportement attendu de notre design. Fermer la fenêtre de simulation.

Créer un test bench (VHDL)

Dans Vivado, nous allons tout d'abord créer un test bench afin de simuler notre design.

Qu'est-ce qu'un test bench? Un test bench est un fichier HDL (VHDL, Verilog…) qui, généralement génère des stimuli (entrées, horloges) à une Unité Sous Test (Unit Under Test (UUT) en anglais).

Pour créer le fichier de test bench dans Vivado, cliquez sur "*Add Sources*" dans le " *Flow Navigator* " et sélectionnez "*Add or create simulation sources*".

Figure 8 - Add or create simulation sources

Ensuite, cliquez sur "*Create File*", sélectionnez VHDL, entrez "*tb_top_level*" comme nom de fichier et cliquez sur "*finish*". Généralement, les tests benches n'ont pas d'entrées ni des sorties, donc si la fenêtre "*Define Module*" apparait, supprimez la première ligne en la sélectionnant et en cliquant sur le bouton "-". Cliquez ensuite sur "*OK*".

Nous pouvons voir notre fichier sous "*Simulation Sources*" dans la fenêtre "*Sources*".

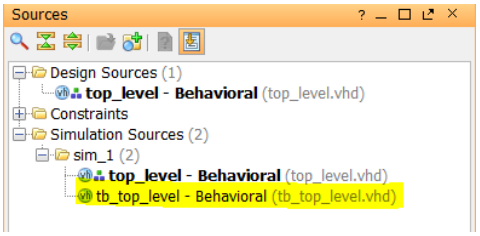

Figure 9 - Simulation Sources dans la fenêtre Sources

Modifier le test bench

Double cliquez sur le fichier dans la fenêtre "*Sources*" pour l'ouvrir dans l'éditeur de texte. On remarque que Vivado a déjà créé un modèle pour notre fichier.

1. Declarer le bloc top_level

Premièrement, déclarez le bloc top_level dans la partie déclarative (entre les mots clés "architecture" et "begin"):

```
38  architecture Behavioral of tb_top_level is
39
40  component top_level is
41      Port (
42          A : in STD_LOGIC;
43          B : in STD_LOGIC;
44          C : in STD_LOGIC;
45          D : in STD_LOGIC;
46          O : out STD_LOGIC
47      );
48  end component top_level;
49
50  begin
```

Figure 10 – Declaration du bloc top_level

2. Declarer les signaux

Ensuite, déclarez les signaux que nous allons utiliser. Nous allons utiliser toutes les entrées et les sorties du bloc top_level donc déclarez ces signaux dans la partie déclarative et initialiser les entrées à '0':

```
49
50      signal A : STD_LOGIC := '0';
51      signal B : STD_LOGIC := '0';
52      signal C : STD_LOGIC := '0';
53      signal D : STD_LOGIC := '0';
54      signal O : STD_LOGIC;
55
56  begin
```

Figure 11 - Signals declaration

3. Instantier l'Unit Under Test (UUT)

Ensuite, instanciez (connectez) le bloc top_level, que nous allons appeler UUT, après le mot clé the "begin" en connectant les signaux que nous avons créé précédemment:

```
58      UUT : top_level
59      Port map(
60          A => A,
61          B => B,
62          C => C,
63          D => D,
64          O => O
65      );
```

Figure 12 - Instancier l'UUT

4. Générer les stimuli

Ensuite, nous allons attribuer une valeur à nos signaux. Nous allons faire basculer nos signaux (alterner leur valeur entre '1' et '0'). A toutes les 10 ns, B toutes les 20 ns, C toutes les 40 ns et D toutes les 80 ns. Toutes les combinaisons seront testées après 160ns (simulation time).

```
66
67      A <= not A after 10 ns;
68      B <= not B after 20 ns;
69      C <= not C after 40 ns;
70      D <= not D after 80 ns;
71
72  end Behavioral;
```

Figure 13 – génération des stimuli

Lancer la simulation avec le test bench

Lancer la simulation en cliquant sur "*Run Simulation > Run Behavioral Simulation*" sous "*Simulation"* dans le "*Flow Navigator*".

Nous pouvons observer la même "*waveform window*" que dans la première partie du tutoriel.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_top_level is
--  Port ( );
end tb_top_level;

architecture Behavioral of tb_top_level is

component top_level is
    Port (
        A : in STD_LOGIC;
        B : in STD_LOGIC;
        C : in STD_LOGIC;
        D : in STD_LOGIC;
        O : out STD_LOGIC
    );
end component top_level;

    signal A : STD_LOGIC := 0;
    signal B : STD_LOGIC := 0;
    signal C : STD_LOGIC := 0;
    signal D : STD_LOGIC := 0;
    signal O : STD_LOGIC;

begin
    UUT : top_level
    Port map(
        A => A,
        B => B,
        C => C,
        D => D,
        O => O
    );

    A <= not A after 10 ns;
    B <= not B after 20 ns;
    C <= not C after 40 ns;
    D <= not D after 80 ns;

end Behavioral;
```