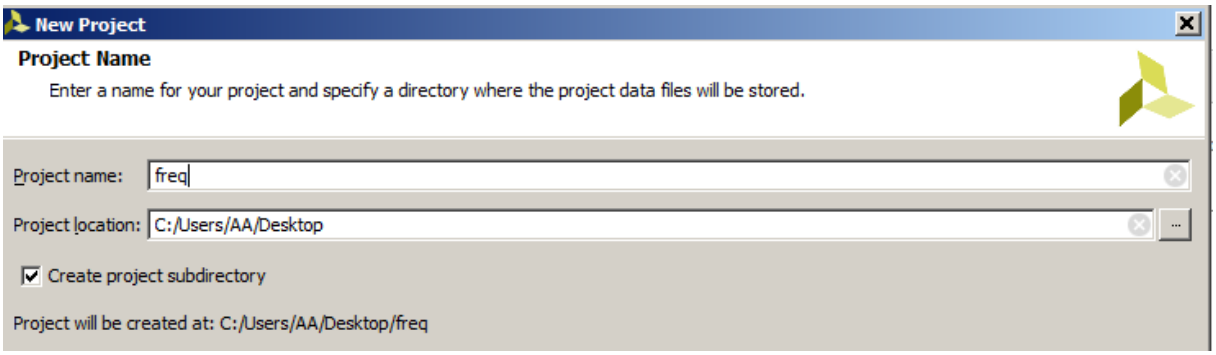
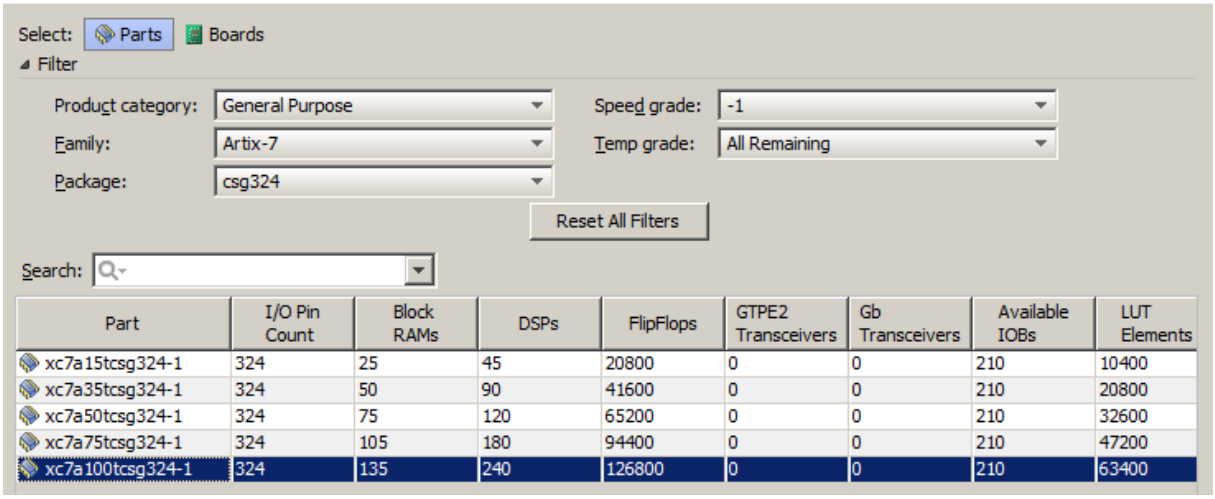
Ouvrir l'outil « Vivado » :

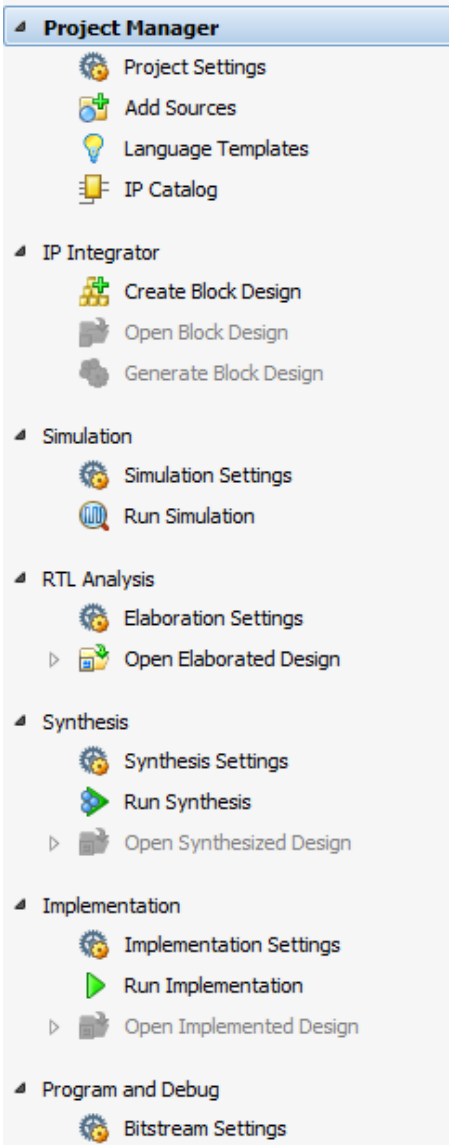
Choisir le nom du projet :



Sélectionner le composant cible :

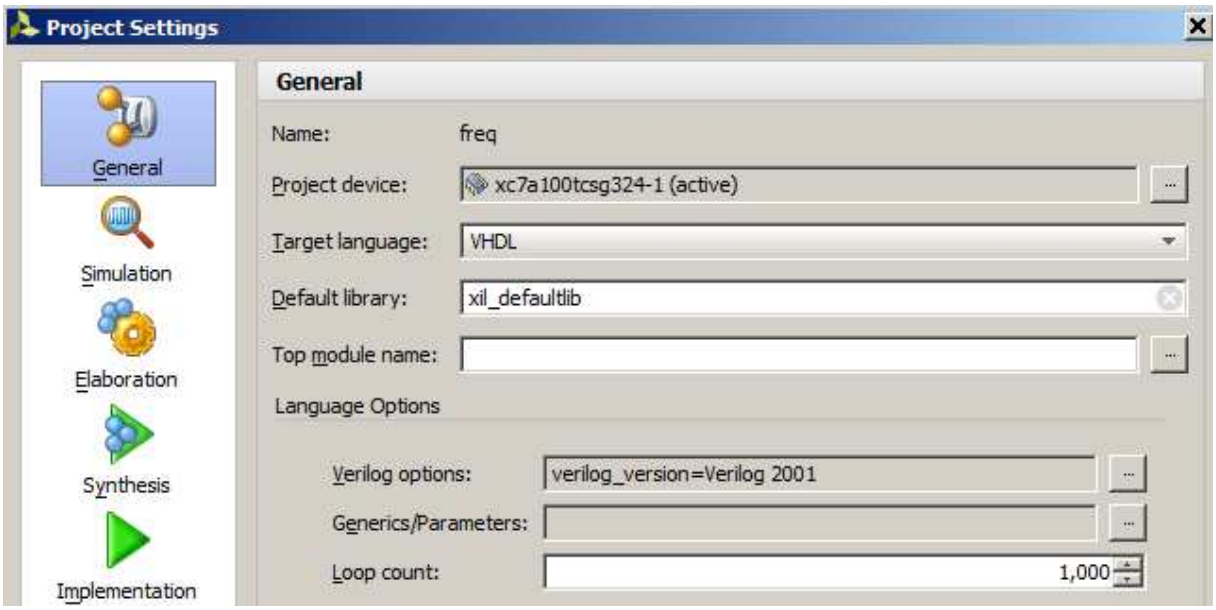


Part	I/O Pin Count	Block RAMs	DSPs	FlipFlops	GTPE2 Transceivers	Gb Transceivers	Available IOBs	LUT Elements
xc7a15tcsg324-1	324	25	45	20800	0	0	210	10400
xc7a35tcsg324-1	324	50	90	41600	0	0	210	20800
xc7a50tcsg324-1	324	75	120	65200	0	0	210	32600
xc7a75tcsg324-1	324	105	180	94400	0	0	210	47200
xc7a100tcsg324-1	324	135	240	126800	0	0	210	63400

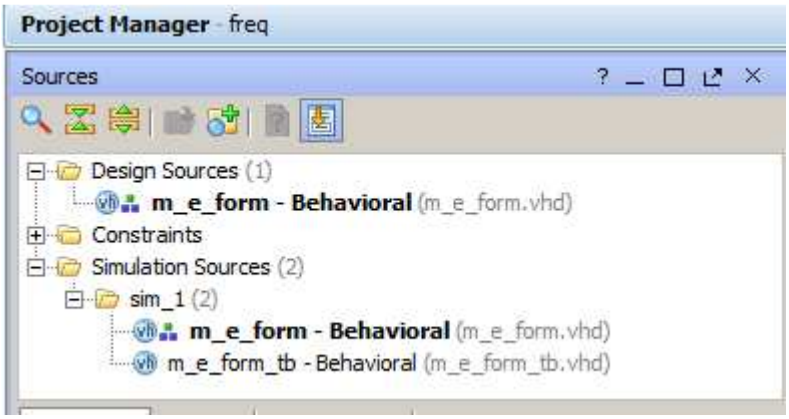


La fenêtre de gauche liste les différentes à franchir pour arriver à la programmation du composant

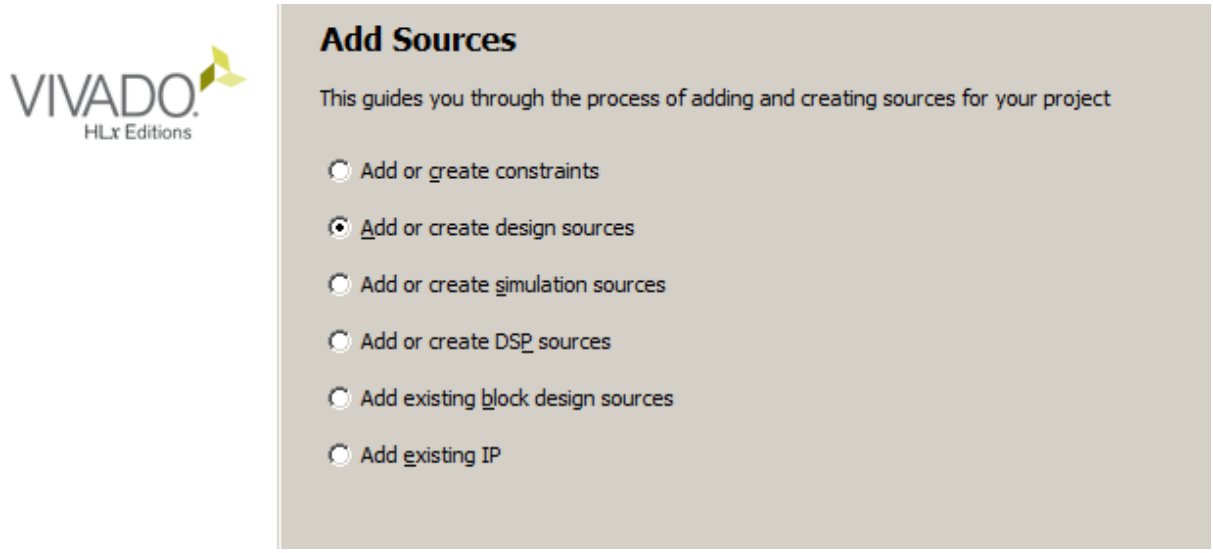
<u>Choisir le langage VHDL</u> :



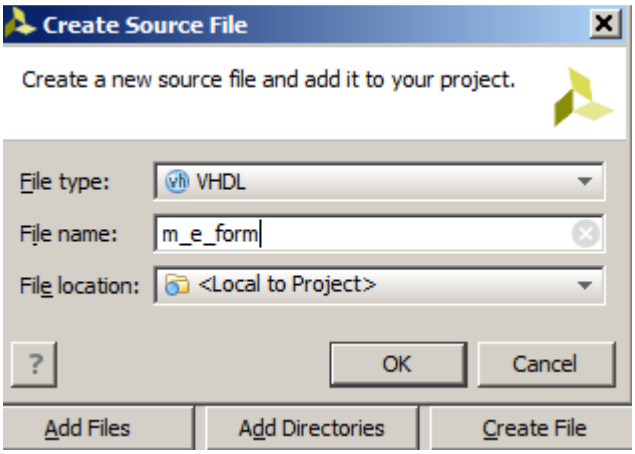
Créer une nouvelle source dans la fenêtre « Sources » :



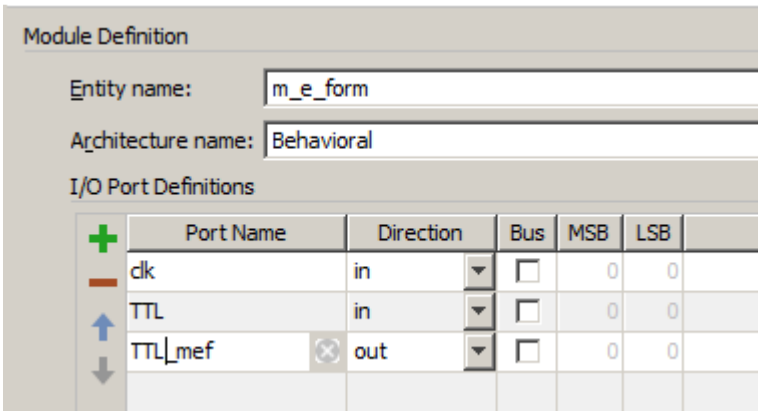
Choisir « design source » pour le type :



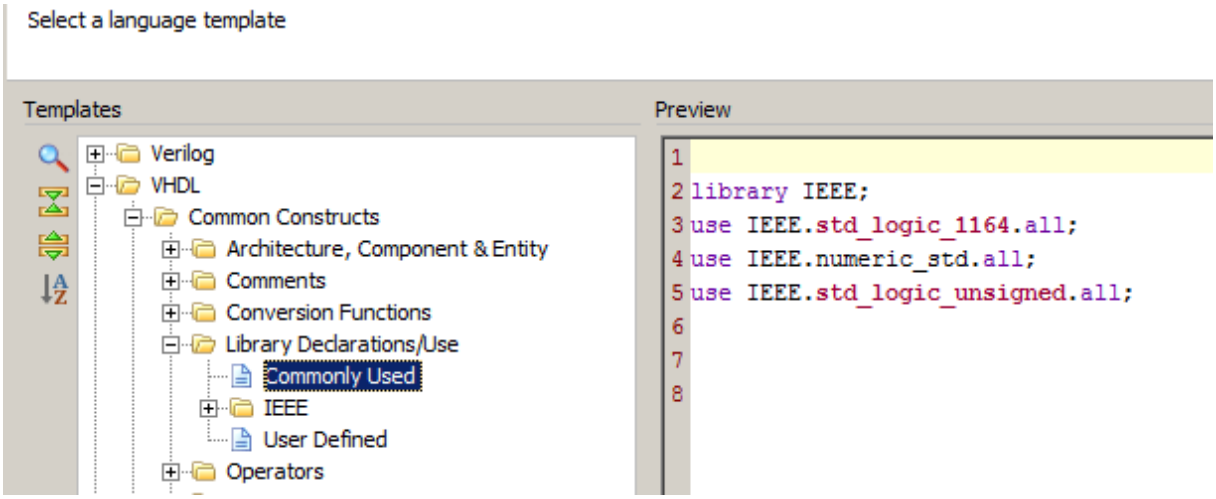
Lui donner un nom :



Lister ses entrées-sorties :



Ajouter les 3 librairies fondamentales en utilisant l'aide :

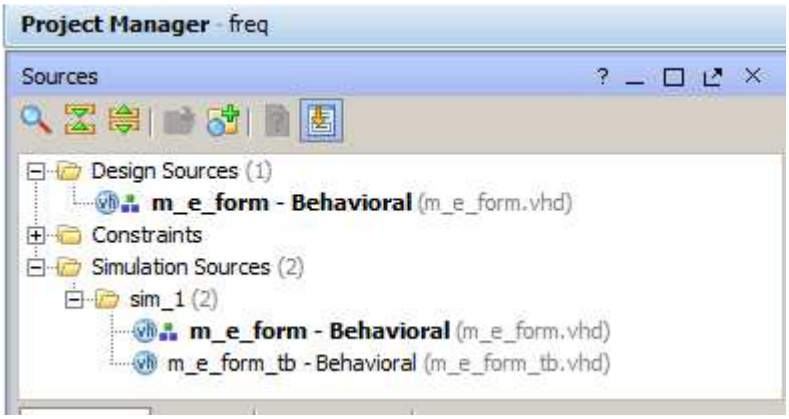


Compléter le programme du module « m_e_form » :

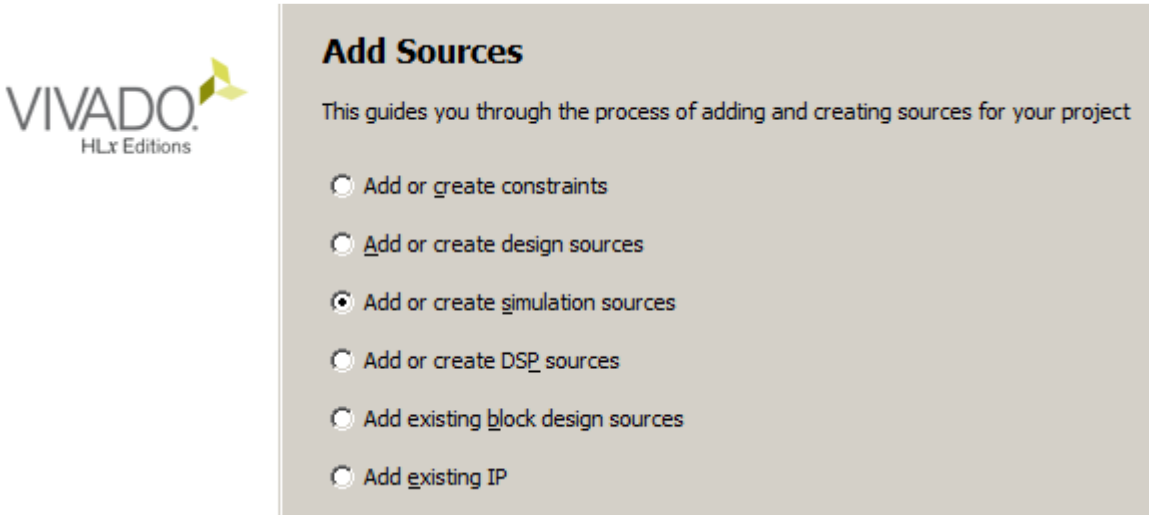
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;

entity m_e_form is
    Port ( clk : in STD_LOGIC;
           TTL : in STD_LOGIC;
           TTL_mef : out STD_LOGIC);
end m_e_form;

architecture Behavioral of m_e_form is
    type etat_type is (debut, mef, fin);
    signal etat : etat_type := debut;
    signal etat_suiv : etat_type ;

begin

process(clk)
begin
    if rising_edge(clk) then
        etat <= etat_suiv;
    end if;
end process;
```

```
process(etat, ttl)
begin
    case etat is
            when debut =>
                        ttl_mef <= '0';
                        if ttl='1' then
                            etat_suiv <= mef;
                        else
                            etat_suiv <= debut;
                        end if;
            when mef =>
                        ttl_mef <= '1';
                        etat_suiv <= fin;
            when fin =>
                        ttl_mef <= '0';
                        if ttl='0' then
                            etat_suiv <= debut;
                        else
                            etat_suiv <= fin;
                        end if;
    end case;
end process;

end Behavioral;
```

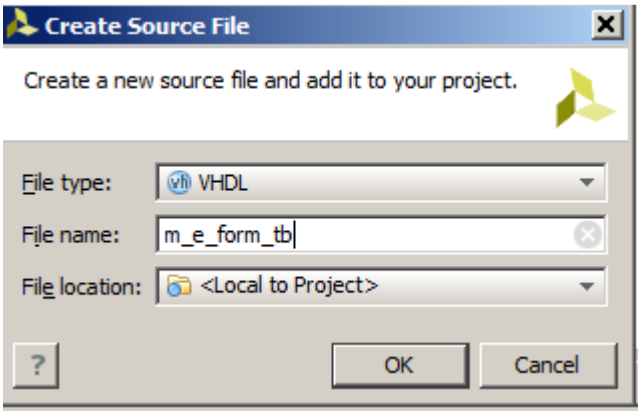
Créer une nouvelle source de type « test bench » pour simuler « m_e_form » :



Choisir le type « simulation » :



Lui donner un nom parlant :

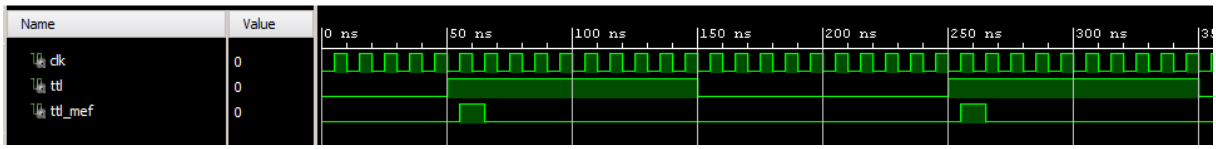


Compléter le fichier de simulation :

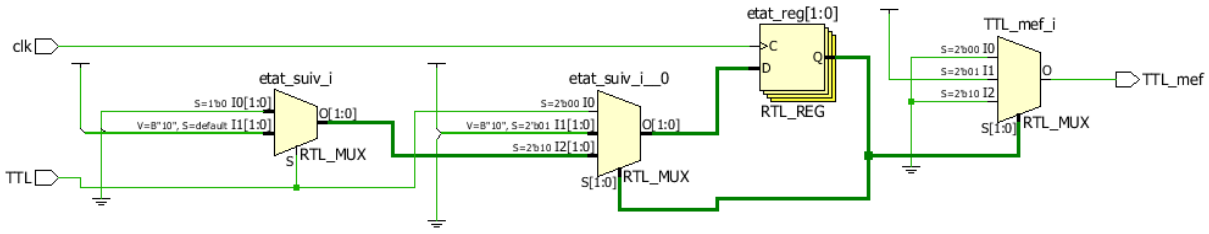
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity m_e_form_tb is
end m_e_form_tb;

architecture Behavioral of m_e_form_tb is
    -- déclaration du module à simuler
    component m_e_form is
        Port ( clk : in STD_LOGIC;
               ttl : in STD_LOGIC;
               ttl_mef : out STD_LOGIC);
    end component;
    -- délaration des signaux d'entrées
    signal clk : std_logic := '0';
    signal ttl : std_logic := '0';
    -- délaration des signaux de sorties
    signal ttl_mef : std_logic;

begin
    -- Connecter le module m_e_form (uut = unit under test)
     uut : m_e_form PORT MAP (
        clk => clk,
        ttl => ttl,
        ttl_mef => ttl_mef);
```

```
    -- horloge 100 MHz : T = 10 ns
    clk_process :process
    begin
       clk <= '0';
       wait for 5 ns;
       clk <= '1';
       wait for 5 ns;
    end process;

    -- stimuli en entrée
    process
    begin
        ttl <= '0';

        wait for 50 ns;

        for i in 0 to 9 loop
        ttl <= '1';
        wait for 100 ns;

        ttl <= '0';
        wait for 100 ns;
        end loop;

        wait;   -- attente infinie
    end process;

end Behavioral;
```

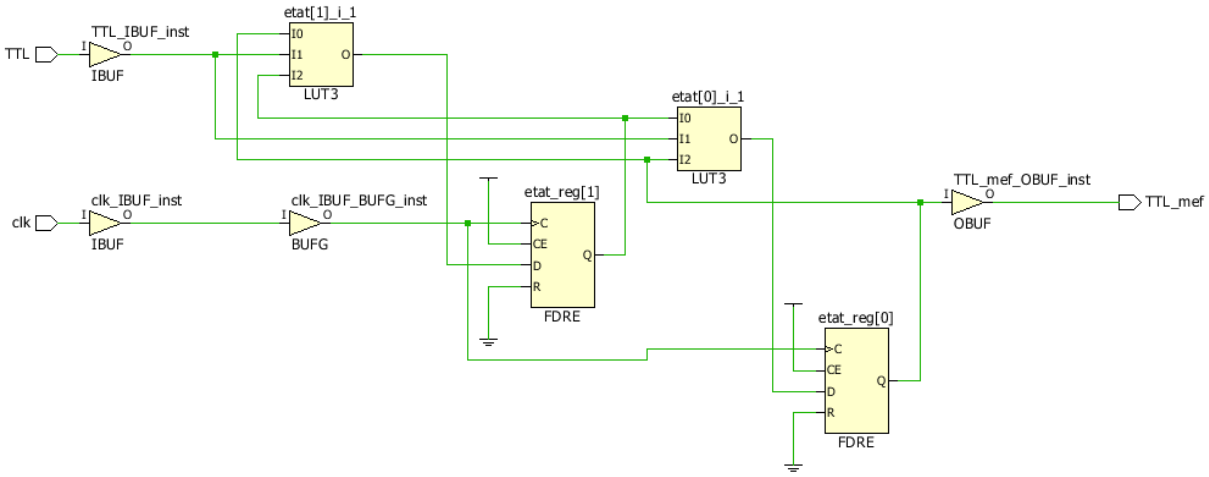
Lancer la simulation pour vérifier le bon fonctionnement :



Ouvrir le schéma RTL : **RTL Analysis >> Schematic**



Ouvrir le schéma post synthèse : **Synthesis >> Open Synthesized Design >> Schematic**



Comparer les 2 schémas !