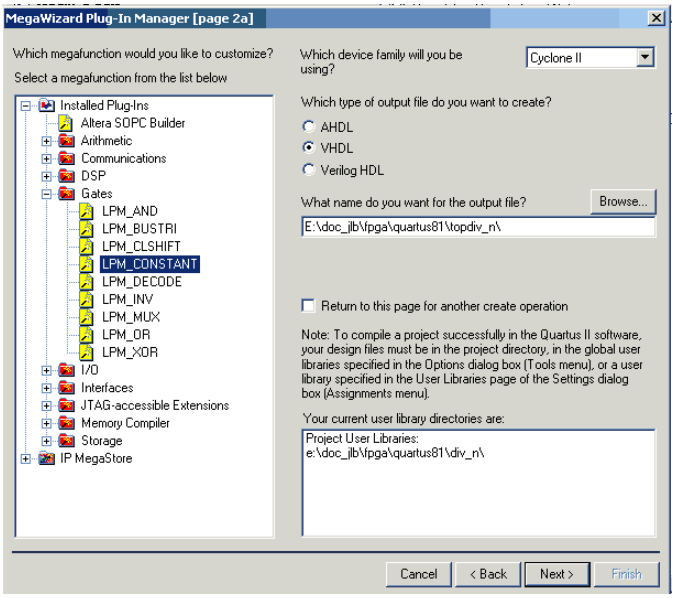
Utilisation du “**In system memory content editor”**

Le « **In system memory content editor** » est un petit utilitaire embarqué dans le circuit FPGA qui permet via le port JTAG de venir modifier à la volée des valeurs de constantes ou des contenus de mémoires (lire et/ou écrire). **Il ne peut donc fonctionner qu’avec une carte cible FPGA.**

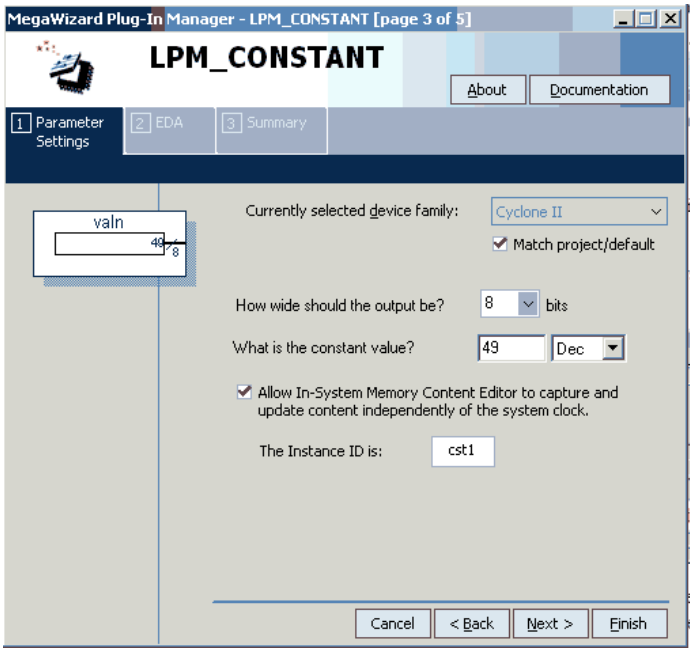
Exemple (projet topdiv_n):

On se propose de créer un compteur/diviseur par n appelé topdiv_n dont la valeur de n peut être modifiée dynamiquement à partir du PC.

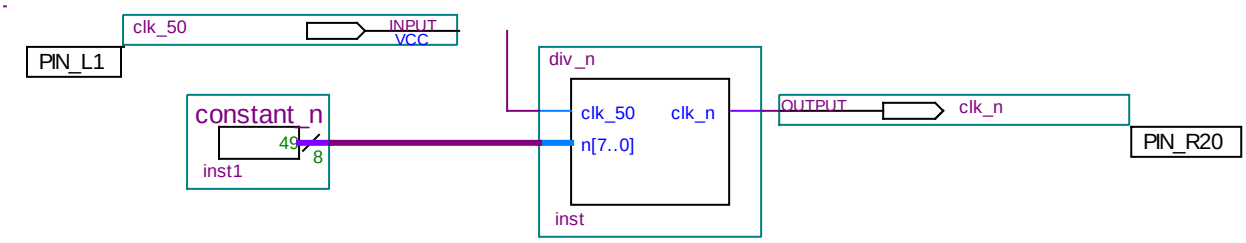
1) Pour cela on crée un composant compteur/diviseur par n appelé div_n (en vhdl ou autre) dont on intègre le symbole dans une feuille type bdf. Pour faire varier la valeur de pré-division n on insère le symbole « constant_n » à partir de **Tools => MegaWizard Plug in manager :**



On définit les caractéristiques de la constante et sa valeur par défaut.



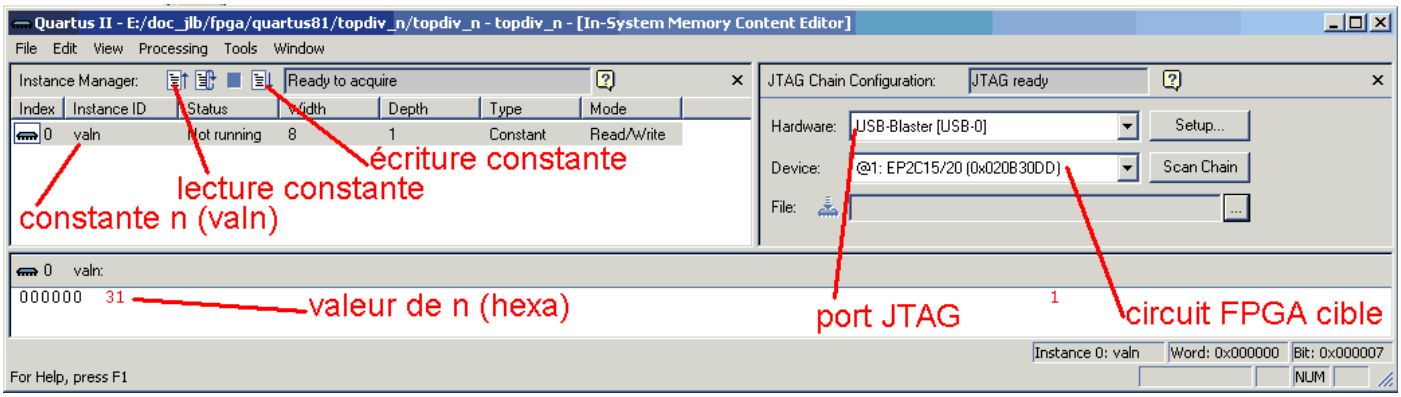
Quartus génère un symbole (.bsf) et un composant associés à cette constante. Il suffit de câbler le symbole obtenu dans la feuille de travail si celle-ci est sous forme de schéma (ou dans le fichier VHDL via des PORTMAP).



2) On affecte ensuite les broches d'entrées-sorties et on compile le projet.

3) On programme le circuit FPGA : celui-ci contient donc le circuit applicatif (le diviseur) + le circuit de gestion de l'interface JTAG et de la constante n.

4) On ouvre dans Quartus le « **In system memory content editor** »: **Tools => In system memory content editor.** On obtient la fenêtre suivante:



En cliquant sur l'icône de lecture on met à jour la valeur de la constante dans la fenêtre de résultat. Inversement en modifiant cette valeur et en cliquant sur l'icône écriture on modifie la valeur de la constante dans le FPGA.

Cette procédure peut également être utilisée pour des mémoires Ram/Rom ce qui peut s'avérer précieux pour la mise au point de circuits (relecture ou modification du contenu).

Ci-dessous le fichier VHDL correspondant au diviseur :

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY div_n IS
        PORT (clk_50 : IN  std_logic;
                        n : in integer range 0 to 255;
                        clk_n : out std_logic
                        );
END div_n;
ARCHITECTURE arch_div_n OF div_n IS
signal s_clk_n : std_logic;
signal s_out : integer range 0 to 255;
BEGIN
        gen_1s: process (clk_50)
        begin
                if clk_50'event and clk_50 ='1' then
```

```
                    if s_out = n then s_out <= 0;
                    s_clk_n <= not s_clk_n ;
                    else  s_out <= s_out + 1;
                    end if;
                end if;
        end process gen_1s;
        clk_n <= s_clk_n;
END arch_div_n;
```