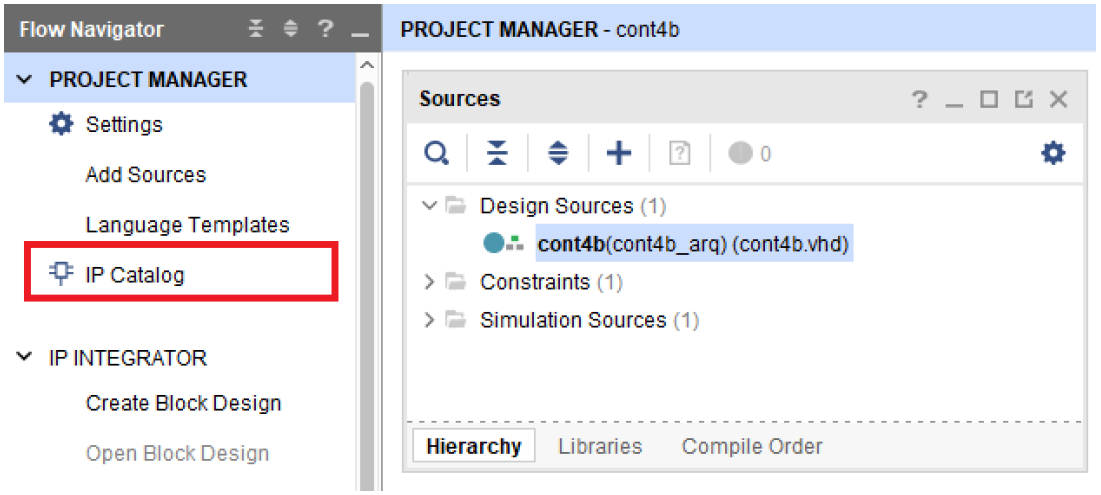
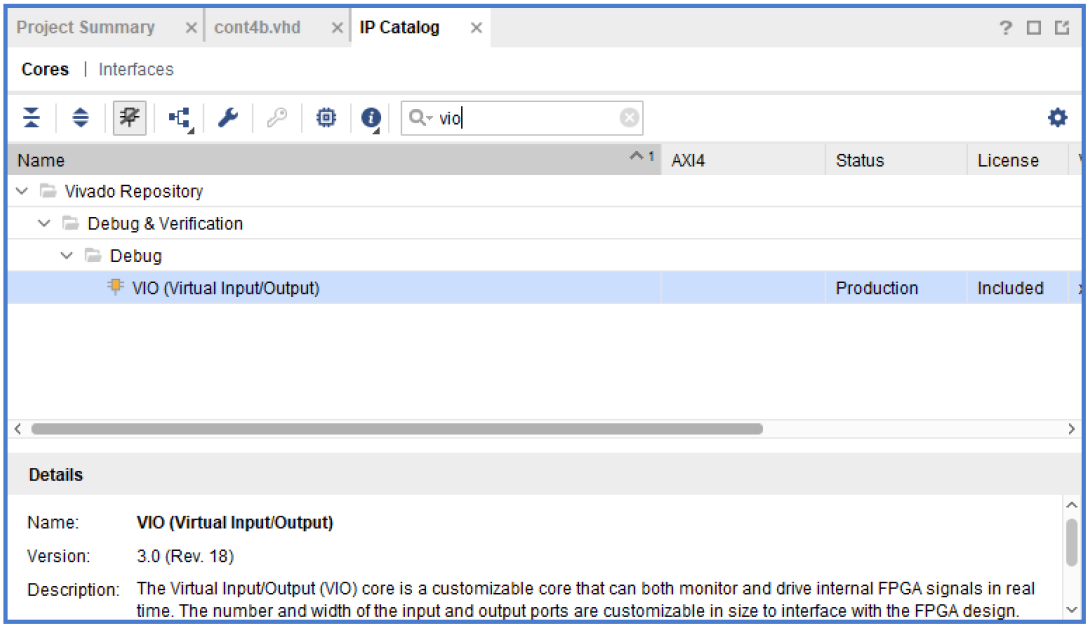
Tutorial para utilizar los bloques VIO e ILA

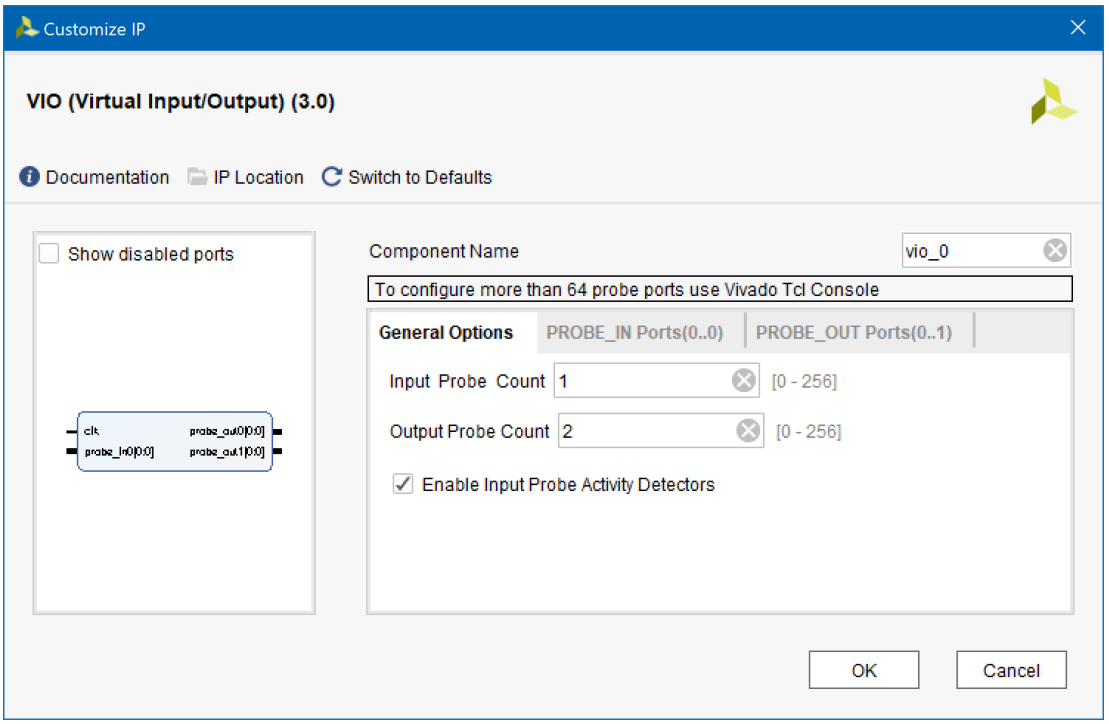
1. Dentro del proyecto seleccionar **IP Catalog**.

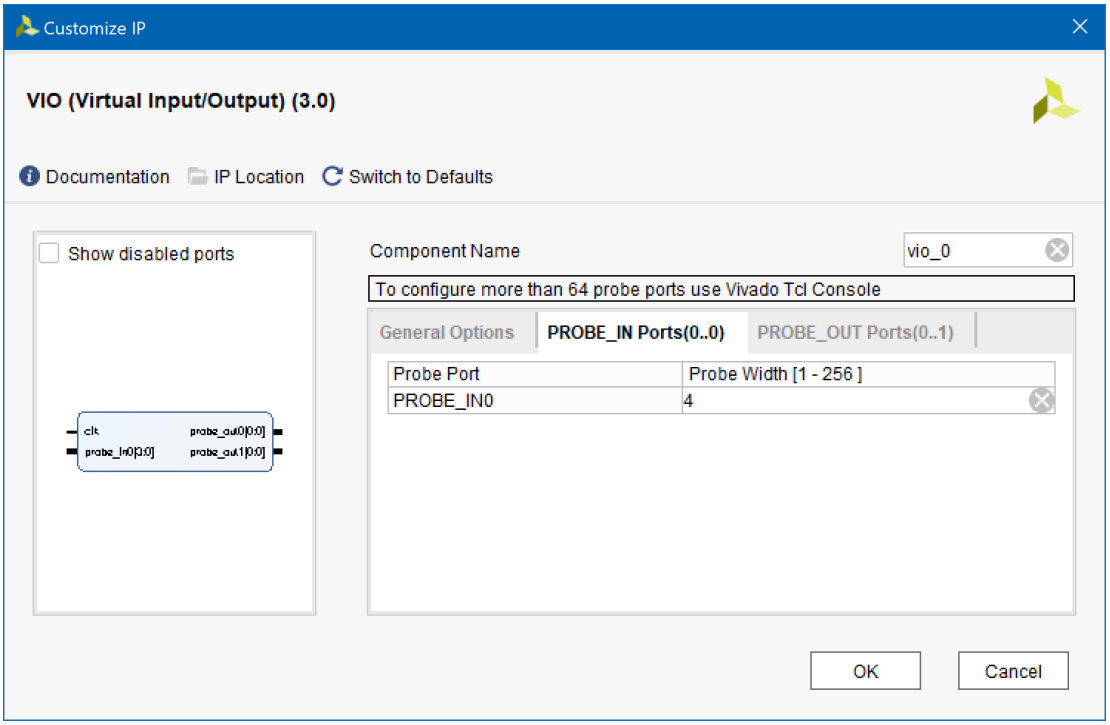


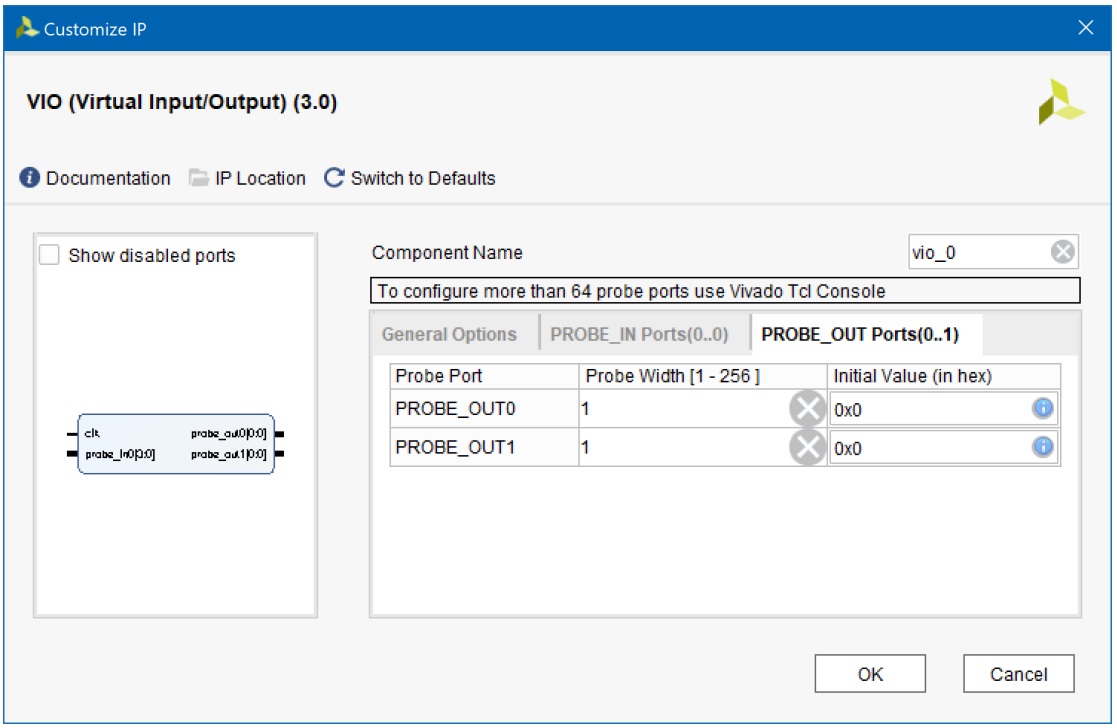
2. En la pestaña **IP Catalog** colocar vio en la casilla de búsqueda y hacer doble click sobre **VIO (Virtual Input/Output)**.



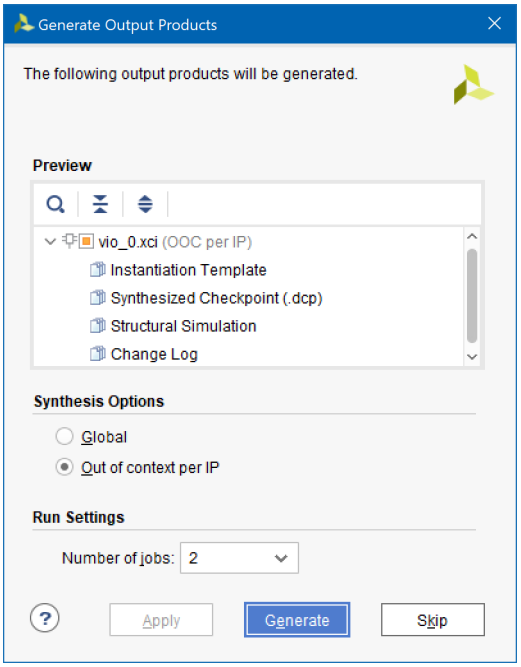
3. Establecer la cantidad de puntas de prueba a utilizar y su tamaño (en el ejemplo actual se está analizando un contador con entrada de reset y habilitación y salida de 4 bits por lo que el VIO deberá contar con una entrada de 4 bits y dos salidas de 1 bit). Presionar **OK** por dos veces.





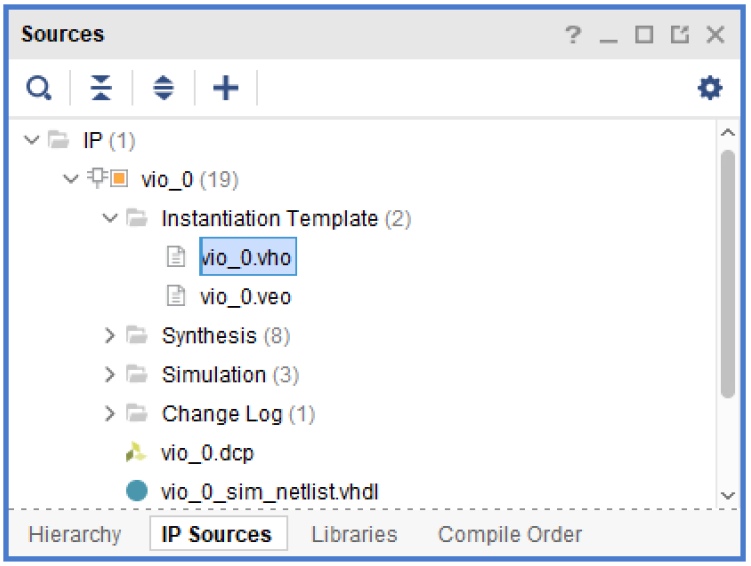


4. Aparecerá la ventana **Generate Output Products**. Presionar el botón **Generate**.

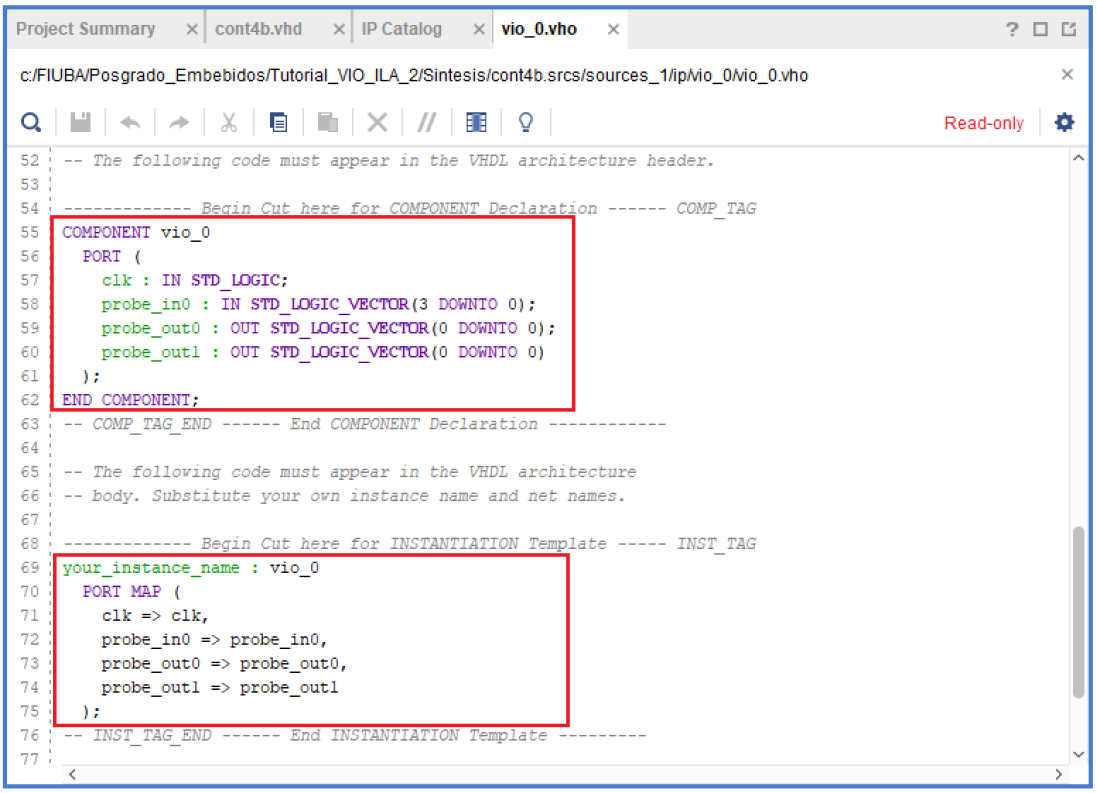


5. Presionar **OK** en la nueva ventana emergente. En este momento comienza la síntesis del bloque que se acaba de configurar.

6. Seguidamente se debe incluir el bloque VIO en el diseño propio. En el panel **Sources**, en la pestaña **IP Sources**, desplegar **vio_0** y luego **Instantiation Template**. Hacer doble click sobre **vio_0.vho**.



7. Copiar la plantilla de declaración de componente y pegarla en el top level del diseño propio (parte declarativa). Proceder del mismo modo con la plantilla de instanciación pero en este caso pegarla en la parte descriptiva de la arquitectura.



Project Summary | cont4b.vhd * | IP Catalog | vio_0.vho

C:/FIUBA/Posgrado_Embebidos/Tutorial_VIO_ILA_2/Fuentes/cont4b.vhd

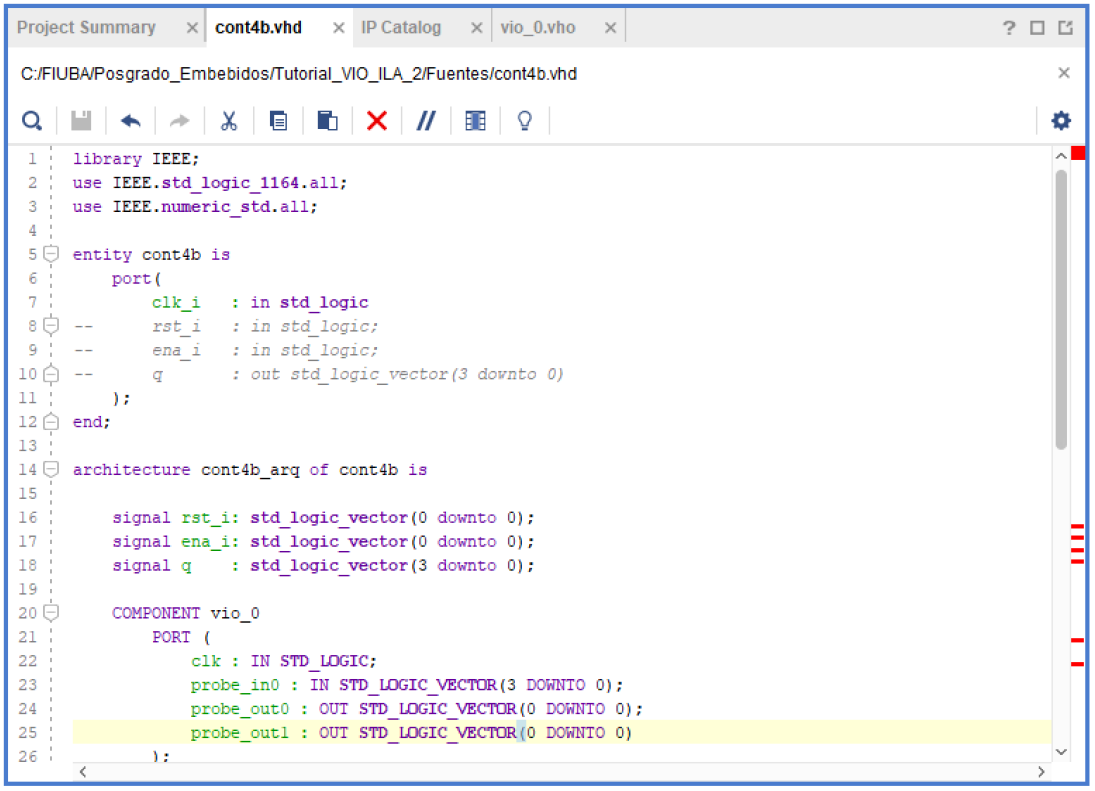
```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  entity cont4b is
6      port(
7          clk_i   : in std_logic;
8          rst_i   : in std_logic;
9          ena_i   : in std_logic;
10         q       : out std_logic_vector(3 downto 0)
11     );
12 end;
13
14 architecture cont4b_arq of cont4b is
15
16     COMPONENT vio_0
17         PORT (
18             clk : IN STD_LOGIC;
19             probe_in0 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
20             probe_out0 : OUT STD_LOGIC_VECTOR(0 DOWNTO 0);
21             probe_out1 : OUT STD_LOGIC_VECTOR(0 DOWNTO 0)
22         );
23     END COMPONENT;
24
25 begin
26
```

Project Summary | cont4b.vhd * | IP Catalog | vio_0.vho

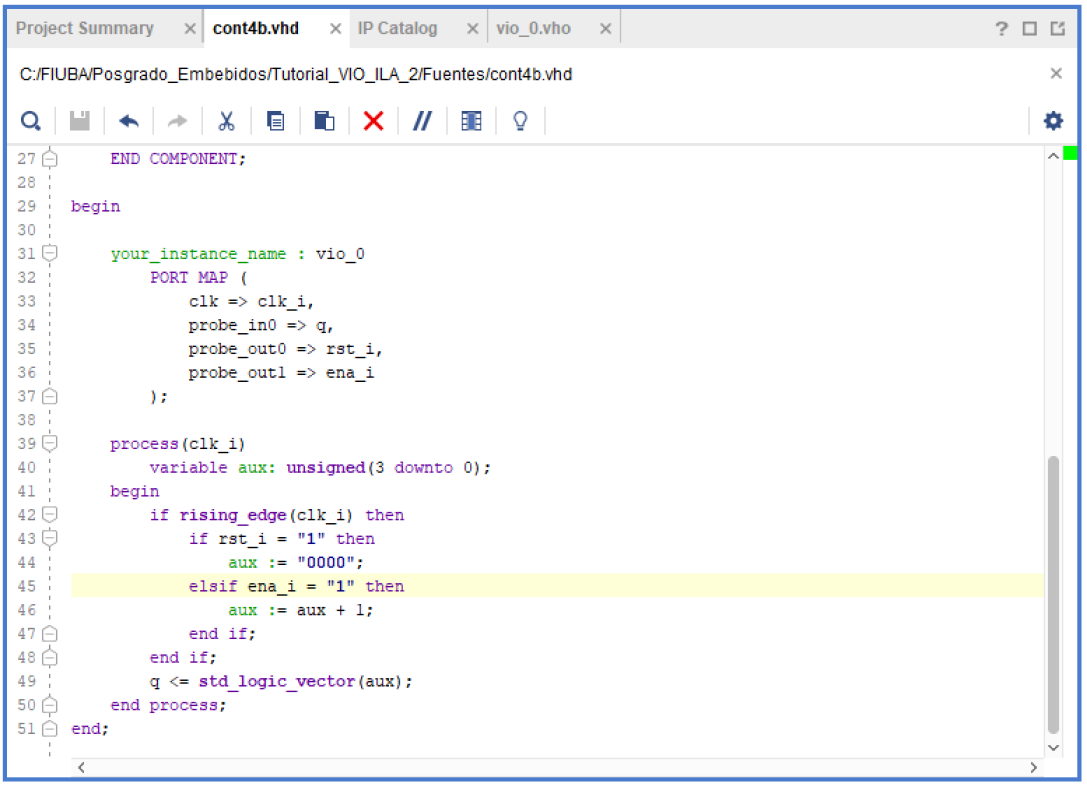
C:/FIUBA/Posgrado_Embebidos/Tutorial_VIO_ILA_2/Fuentes/cont4b.vhd

```
13
14 architecture cont4b_arq of cont4b is
15
16     COMPONENT vio_0
17         PORT (
18             clk : IN STD_LOGIC;
19             probe_in0 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
20             probe_out0 : OUT STD_LOGIC_VECTOR(0 DOWNTO 0);
21             probe_out1 : OUT STD_LOGIC_VECTOR(0 DOWNTO 0)
22         );
23     END COMPONENT;
24
25 begin
26
27     your_instance_name : vio_0
28         PORT MAP (
29             clk => clk,
30             probe_in0 => probe_in0,
31             probe_out0 => probe_out0,
32             probe_out1 => probe_out1
33         );
34
35     process(clk_i)
36         variable aux: unsigned(3 downto 0);
37     begin
38         if rising_edge(clk_i) then
```

8. Comentar todos los puertos del componente menos el de reloj y crear señales para poder utilizar en el bloque VIO como entradas y salidas.



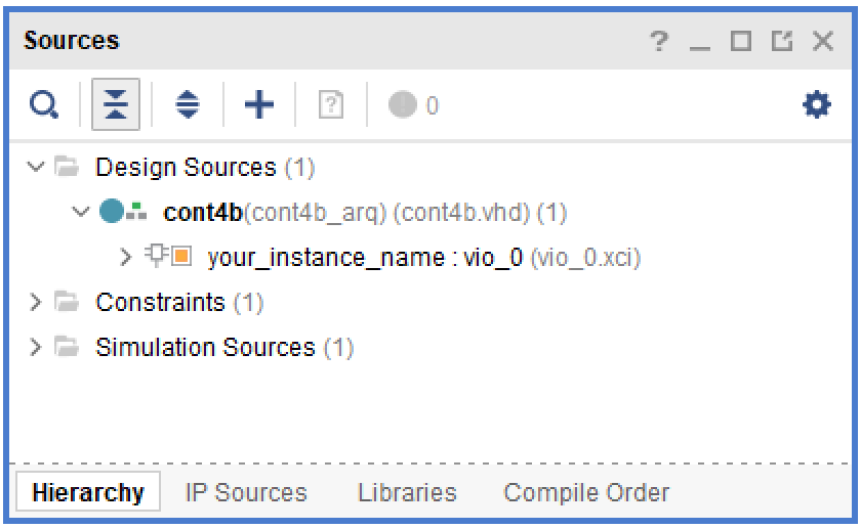
9. Conectar las señales creadas y el reloj al componente VIO. Tener en cuenta que rst_i y ena_i ahora son vectores por lo que debe ser cambiada la forma en que se tratan dentro del process.



10. El código completo debería verse así:

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

entity cont4b is
    port(
        clk_i   : in std_logic
--      rst_i   : in std_logic;
--      ena_i   : in std_logic;
--      q       : out std_logic_vector(3 downto 0)
    );
end;

architecture cont4b_arq of cont4b is

    signal rst_i: std_logic_vector(0 downto 0);
    signal ena_i: std_logic_vector(0 downto 0);
    signal q    : std_logic_vector(3 downto 0);

    COMPONENT vio_0
        PORT (
            clk : IN STD_LOGIC;
            probe_in0 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
            probe_out0 : OUT STD_LOGIC_VECTOR(0 DOWNTO 0);
            probe_out1 : OUT STD_LOGIC_VECTOR(0 DOWNTO 0)
        );
    END COMPONENT;

begin
    your_instance_name : vio_0
        PORT MAP (
            clk => clk_i,
            probe_in0 => q,
            probe_out0 => rst_i,
            probe_out1 => ena_i
        );

    process(clk_i)
        variable aux: unsigned(3 downto 0);
    begin
        if rising_edge(clk_i) then
            if rst_i = "1" then
                aux := "0000";
            elsif ena_i = "1" then
                aux := aux + 1;
            end if;
        end if;
        q <= std_logic_vector(aux);
    end process;
end;
```

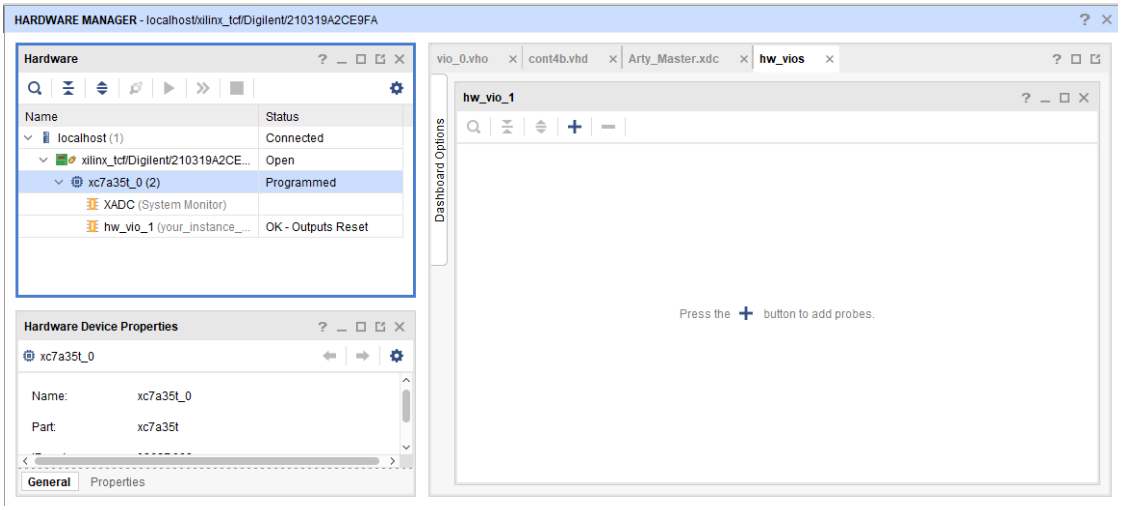
11. Luego de esto debería observarse la siguiente estructura:



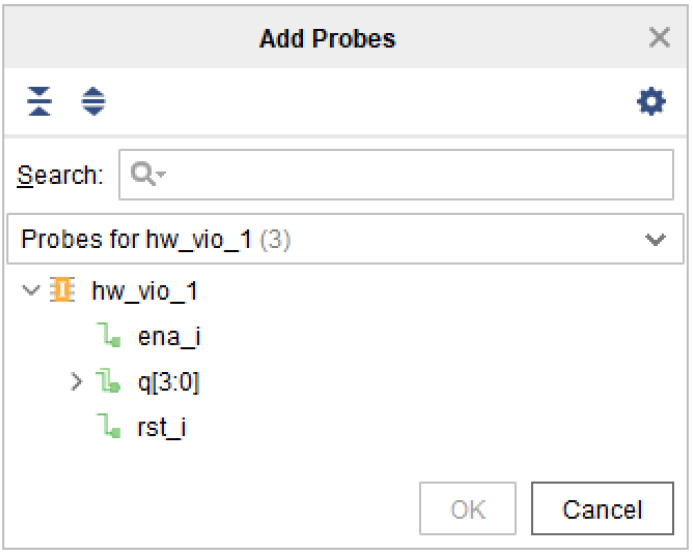
12. Recordar modificar el archivo de restricciones para dejar sólo el reloj.

13. Generar el archivo de configuración (.bit) y abrir el **Hardware Manager**. Conectarse con la FPGA y configurarla.

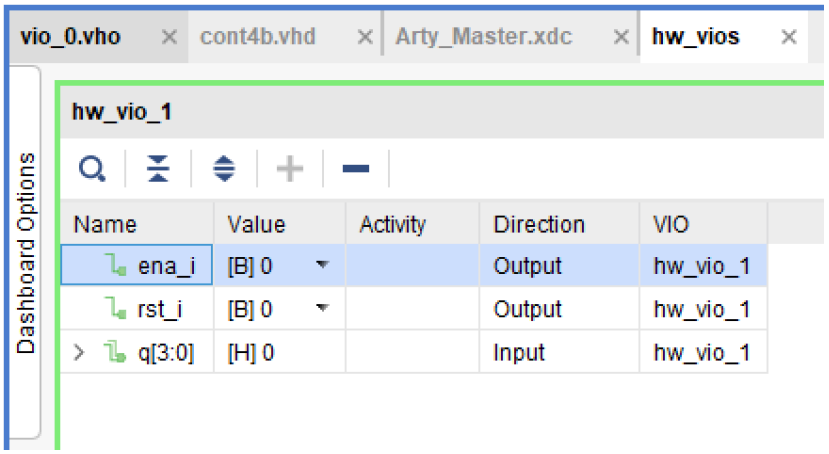
14. Una vez configurada se verá lo siguiente:



15. Presionar el símbolo "**+**" para agregar las señales que se desea comandar y visualizar. Seleccionarlas y presionar **OK**.



16. Luego del paso anterior debería observarse lo siguiente:



A partir de este momento se puede controlar tanto el reset (segunda línea) como el enable (primera línea) y observar la salida en la tercera línea. La figura siguiente muestra un instante en el que la habilitación está en '1', el reset en '0' y la salida exhibe un 0x9.