Cómo instanciar bloques IP en código VHDL-Verilog en Vivado

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2024/06/17/como-instanciar-bloques-ip-en-codigo-vhdl-verilog-en-vivado/

Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

Fecha última modificación: 22/02/2025

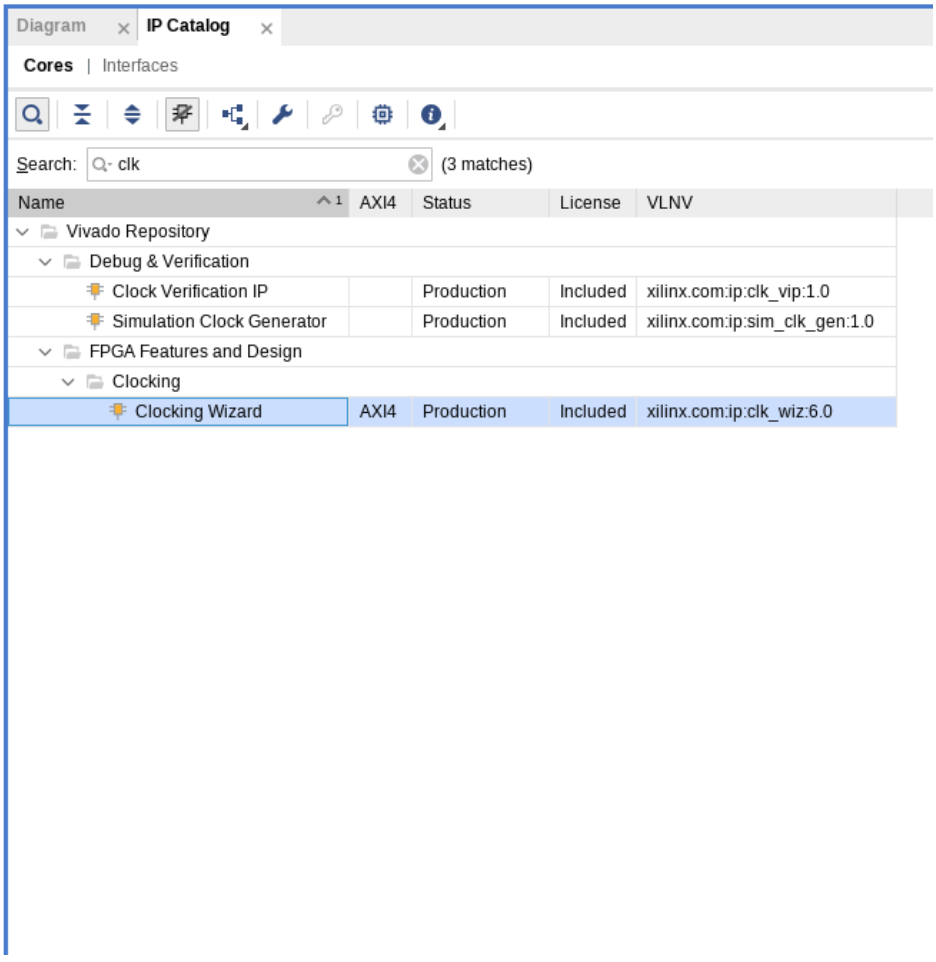
https://soceame.wordpress.com/2024/06/17/como-instanciar-bloques-ip-en-codigo-vhdl-verilog-en-vivado/

Si has usado el Block design de Vivado, habrás visto que puedes colocar bloques IP en un Diagram, y luego el propio Vivado genera el encapsulado de ese *Diagram* y te genera un .vhd o un .v. Bien, pues en esta entrada te voy a explicar cómo insertar un bloque IP directamente en VHDL/Verilog sin usar el Block design.

Pasos

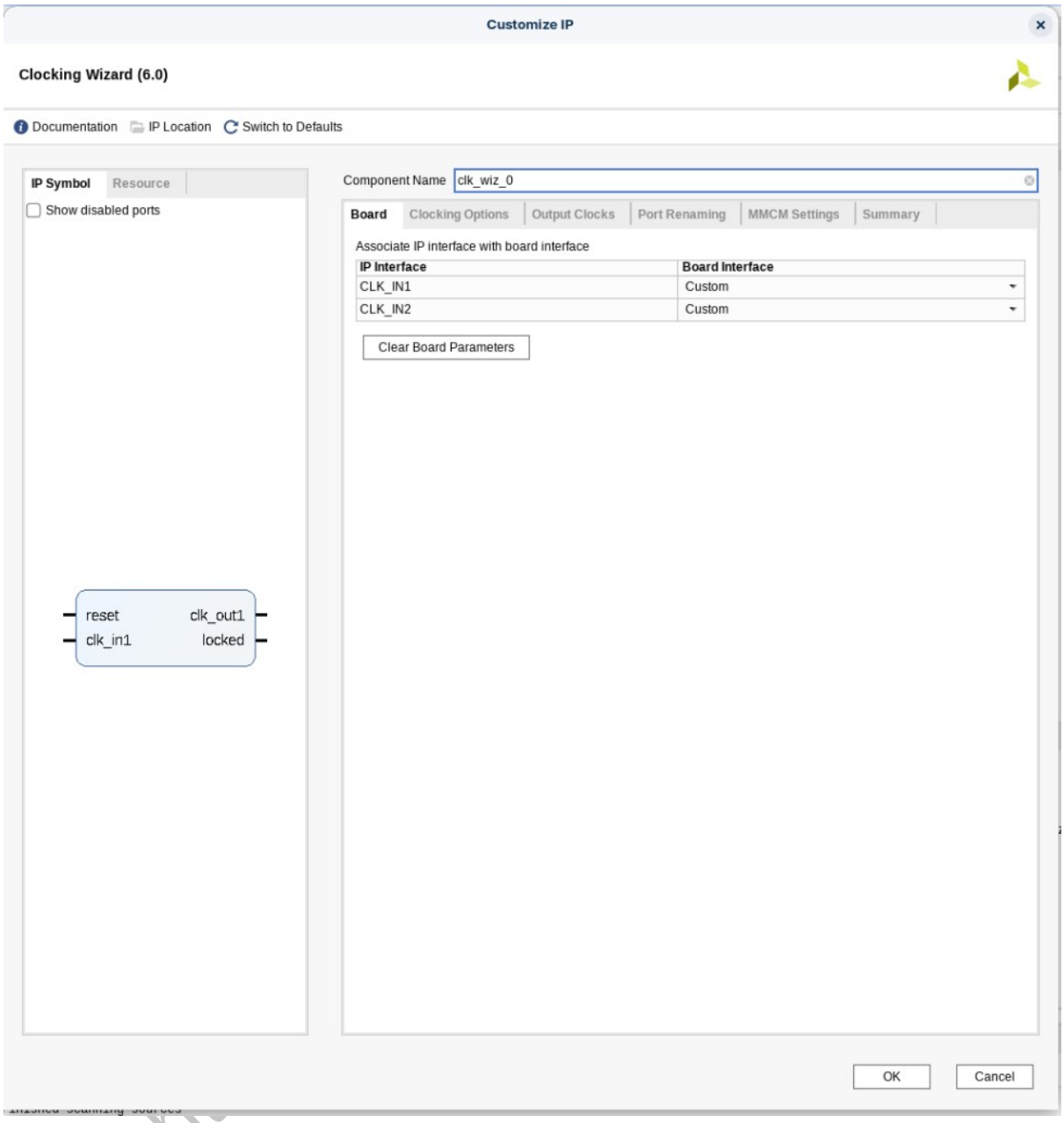
Lo primero que tienes que hacer es abrir el **IP catalog**

Se abre la pestaña. Ahora imagina que quieres insertar un PLL, lo seleccionas.



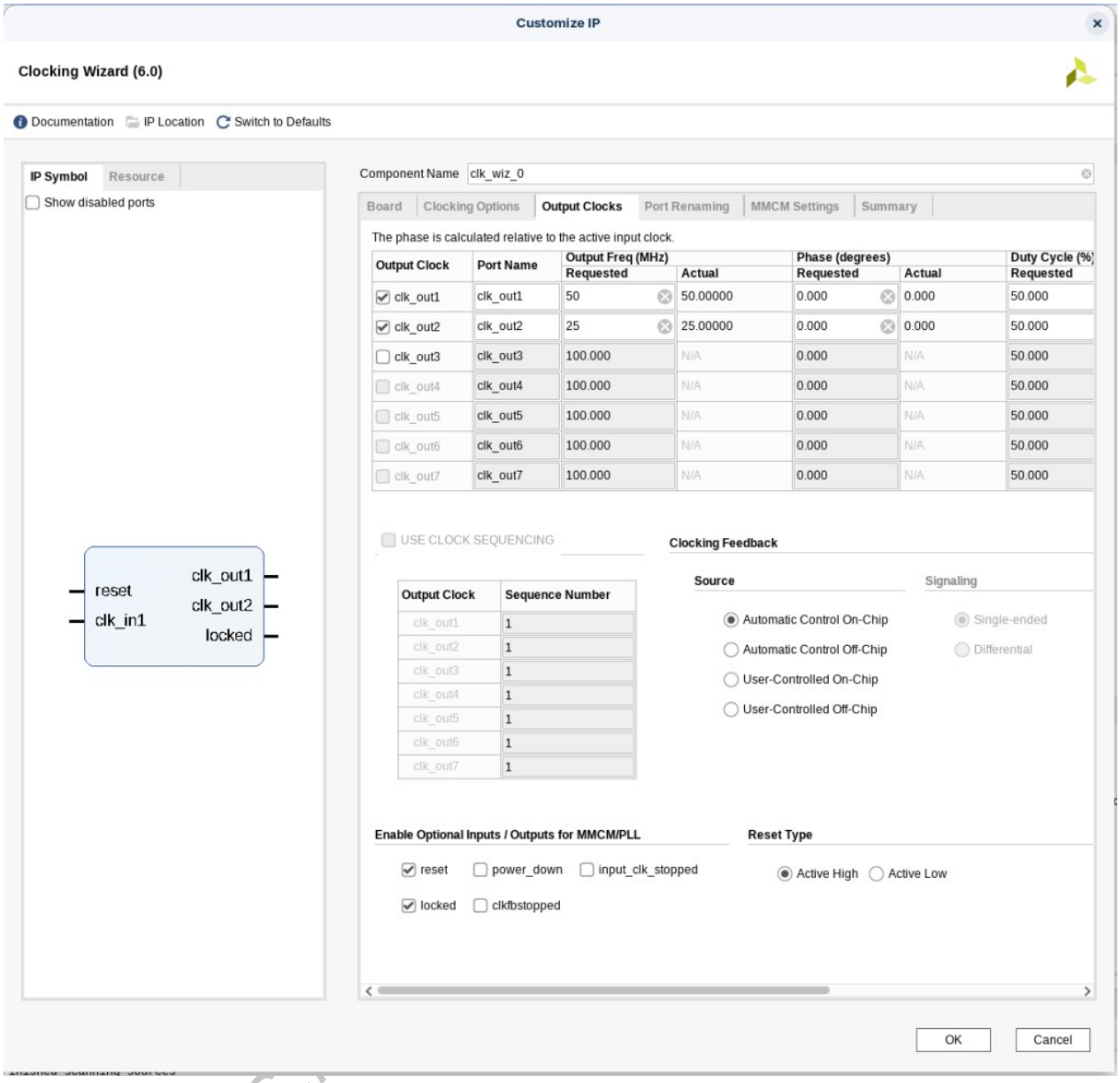
https://soceame.wordpress.com/2024/06/17/como-instanciar-bloques-ip-en-codigo-vhdl-verilog-en-vivado/

Se abre la pestaña del *Clocking Wizard*



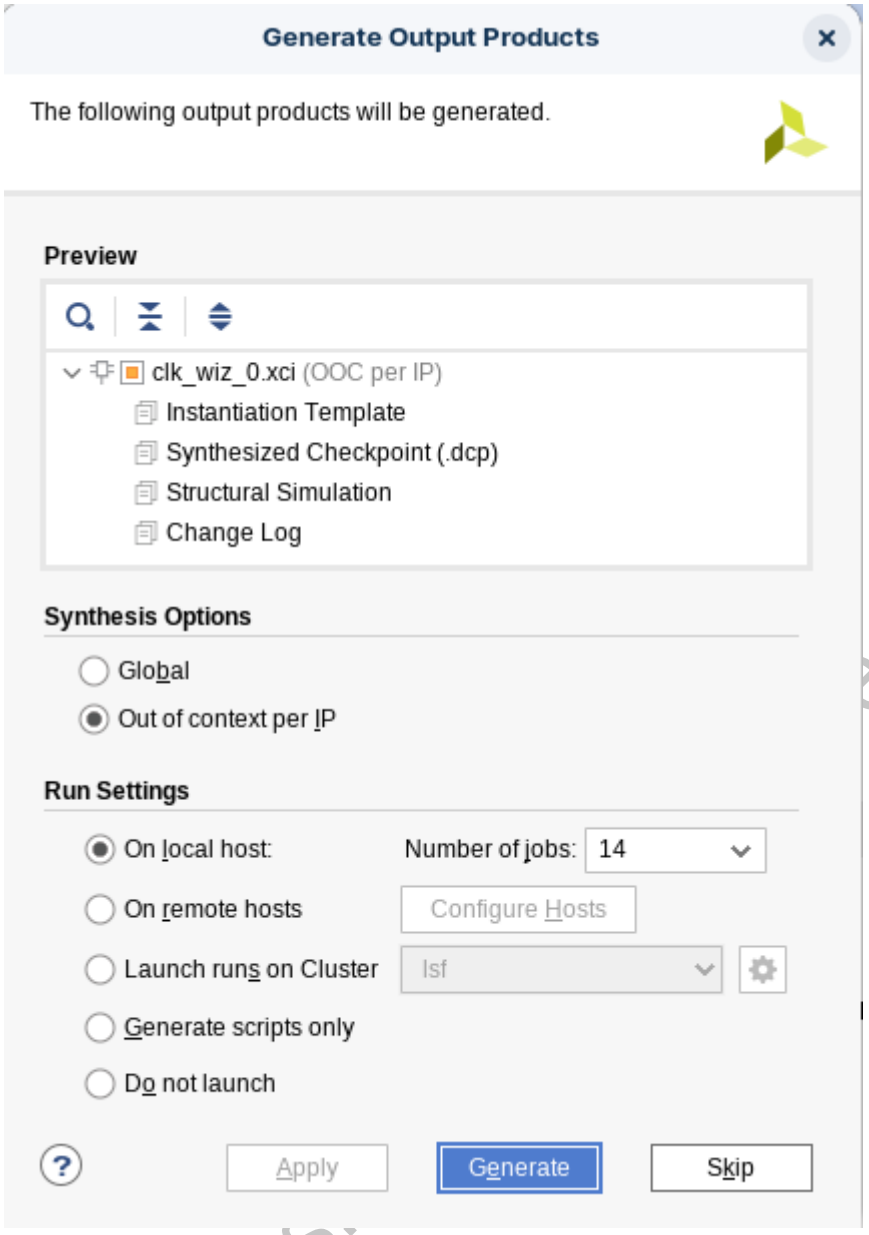
Configuramos el *Clocking wizard*.

https://soceame.wordpress.com/2024/06/17/como-instanciar-bloques-ip-en-codigo-vhdl-verilog-en-vivado/

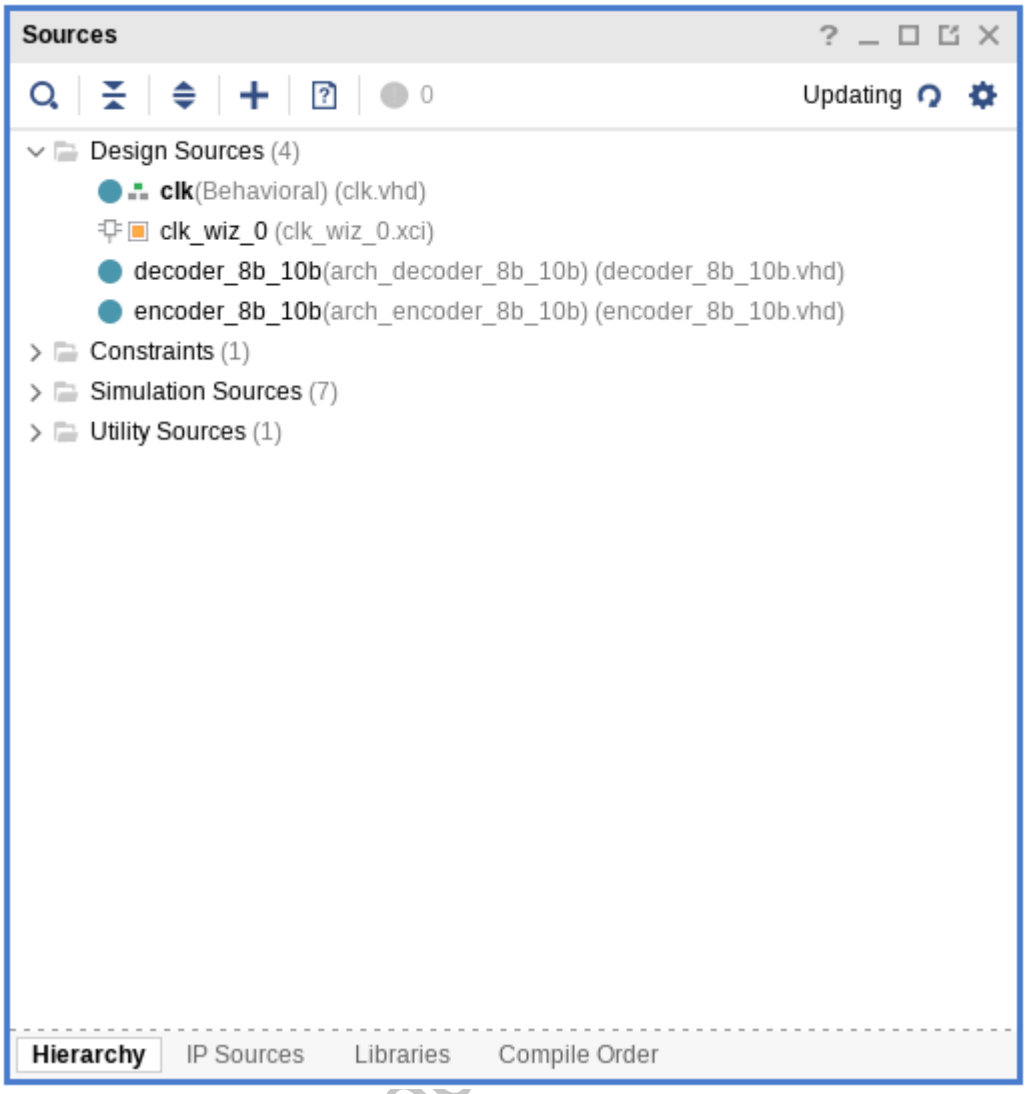


Y una vez lo hemos configurado le decimos que nos genere los **Output Products**

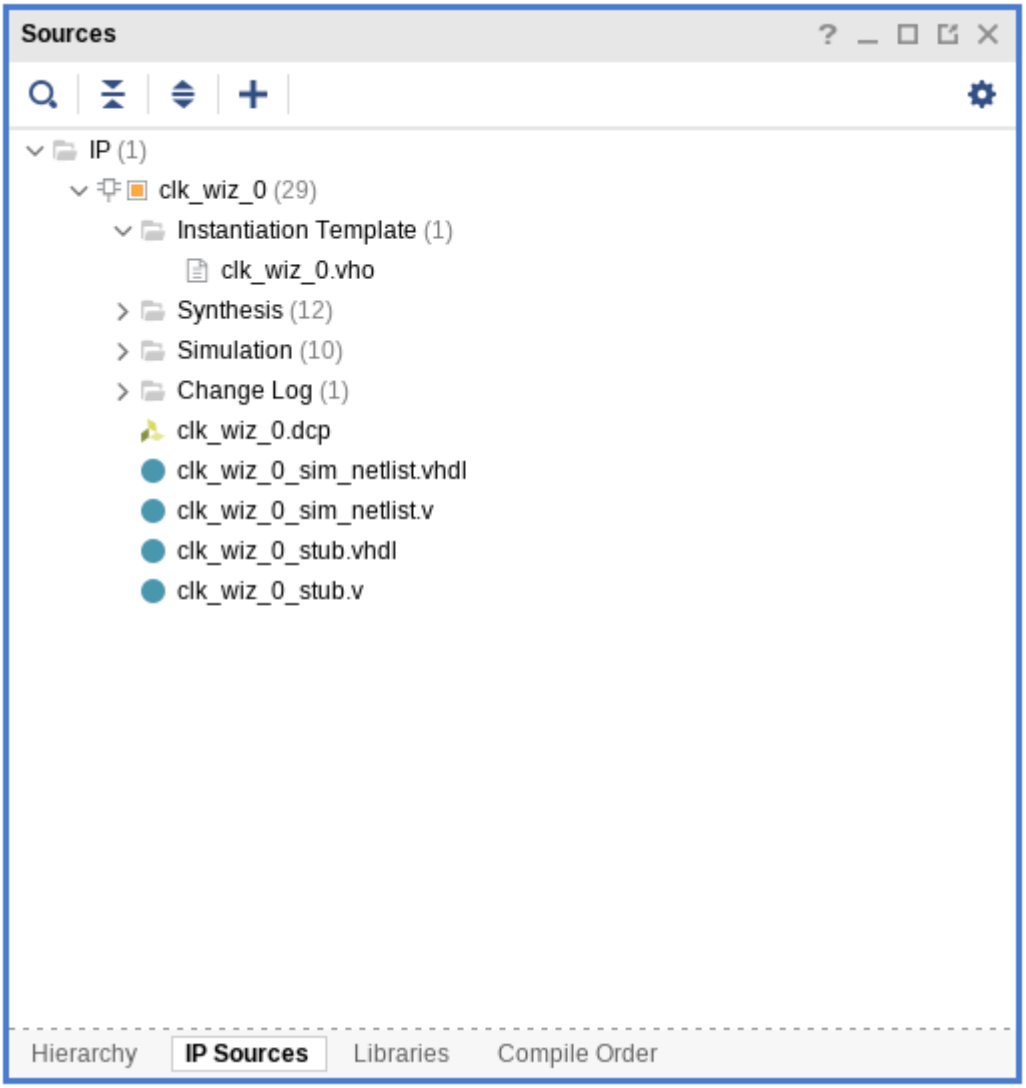
https://soceame.wordpress.com/2024/06/17/como-instanciar-bloques-ip-en-codigo-vhdl-verilog-en-vivado/



Ahora en *Sources* nos aparece el bloque IP configurado con el *clk_wizard*



Bien, pues en la pestaña *IP sources*, aparece el *clk_wizard* con un desplegable que contiene archivos (estos archivos se generan al generar los **Output Products**).



Y en la pestaña **Instantiation Template** aparecen diferentes archivos dependiendo del lenguaje HDL del proyecto creado, si es VHDL aparece un *.vho* y si el Verilog aparece un *.veo.*

Si lo abrimos vemos código para instanciar un bloque IP. Este es el código que instancia el bloque IP en código VHDL.

https://soceame.wordpress.com/2024/06/17/como-instanciar-bloques-ip-en-codigo-vhdl-verilog-en-vivado/

```
------------------------------------------------------------------------------
--  Output     Output      Phase    Duty Cycle   Pk-to-Pk     Phase
--   Clock     Freq (MHz)  (degrees)    (%)     Jitter (ps)  Error (ps)
------------------------------------------------------------------------------
-- clk_out1__50.00000______0.000______50.0______151.636____98.575
-- clk_out2__25.00000______0.000______50.0______175.402____98.575
--
------------------------------------------------------------------------------
-- Input Clock   Freq (MHz)    Input Jitter (UI)
------------------------------------------------------------------------------
-- __primary_________100.000____________0.010


-- The following code must appear in the VHDL architecture header:
------------ Begin Cut here for COMPONENT Declaration ------ COMP_TAG
component clk_wiz_0
port
 (-- Clock in ports
  -- Clock out ports
  clk_out1          : out    std_logic;
  clk_out2          : out    std_logic;
  -- Status and control signals
  reset             : in     std_logic;
  locked            : out    std_logic;
  clk_in1           : in     std_logic
 );
end component;

-- COMP_TAG_END ------ End COMPONENT Declaration ------------
-- The following code must appear in the VHDL architecture
-- body. Substitute your own instance name and net names.
------------ Begin Cut here for INSTANTIATION Template ----- INST_TAG
your_instance_name : clk_wiz_0
   port map (
  -- Clock out ports
   clk_out1 => clk_out1,
   clk_out2 => clk_out2,
  -- Status and control signals
   reset => reset,
   locked => locked,
   -- Clock in ports
   clk_in1 => clk_in1
 );
   INST_TAG_END ------ End INSTANTIATION Template
```

Si lo instanciamos en un fichero, vemos cómo el bloque IP ahora cuelga del fichero

(instanciación)

https://soceame.wordpress.com/2024/06/17/como-instanciar-bloques-ip-en-codigo-vhdl-verilog-en-vivado/

```
architecture Behavioral of clk is
    component clk_wiz_0
    port
     (-- Clock in ports
      -- Clock out ports
      clk_out1          : out    std_logic;
      clk_out2          : out    std_logic;
      -- Status and control signals
      reset             : in     std_logic;
      locked            : out    std_logic;
      clk_in1           : in     std_logic
     );
    end component;
signal clk_out1, clk_out2, reset, locked, clk_in1 : std_logic;

begin

    your_instance_name : clk_wiz_0
       port map (
      -- Clock out ports
       clk_out1 => clk_out1,
       clk_out2 => clk_out2,
      -- Status and control signals
       reset => reset,
       locked => locked,
       -- Clock in ports
       clk_in1 => clk_in1
     );

end Behavioral;
```

Y esta es la forma de instanciar bloques IP. Esto puede ser muy útil con ILAs y VIOs, porque permiten la depuración en real.

Consideración

Si decides instanciar un ILA por ejemplo, y lo que quieres es ver una señal de 1 bit, el *Instantiation Template* genera un *component* como el siguiente.

```
------------- Begin Cut here for COMPONENT Declaration ------ COMP_TAG
COMPONENT ila_0

PORT (
    clk : IN STD_LOGIC;



    probe0 : IN STD_LOGIC_VECTOR(0 DOWNTO 0)
);
END COMPONENT  ;
```

Esto genera un error debido a que no puedes crear un ***std_logic_vector*** de un solo bit, por lo que para solucionarlo tienes que usar un **std_logic** normal, solo modificas el component que instancias.

https://soceame.wordpress.com/2024/06/17/como-instanciar-bloques-ip-en-codigo-vhdl-verilog-en-vivado/

```
COMPONENT ila_0

PORT (
    clk : IN STD_LOGIC;


    probe0 : IN STD_LOGIC
);
END COMPONENT  ;
```