Cómo añadir un PLL en Quartus

Creador: David Rubio G.

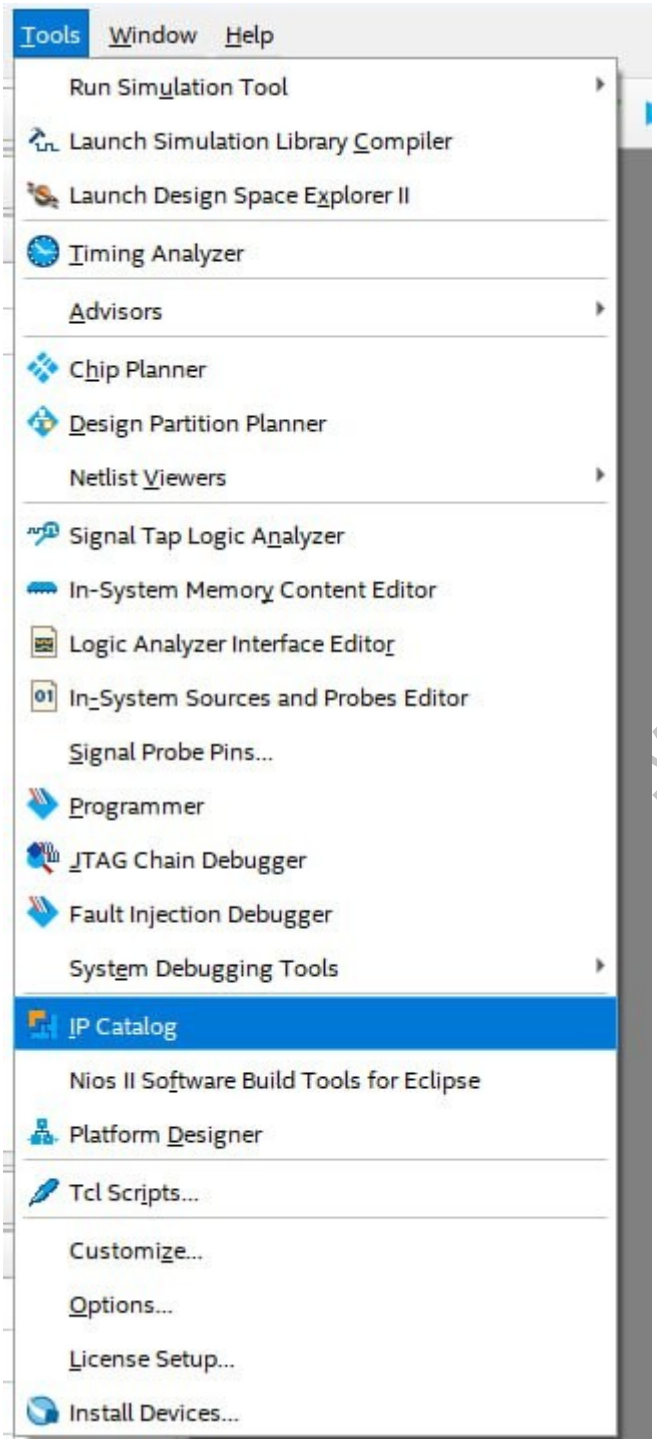
Entrada: https://soceame.wordpress.com/2025/01/14/como-anadir-un-pll-en-quartus/

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

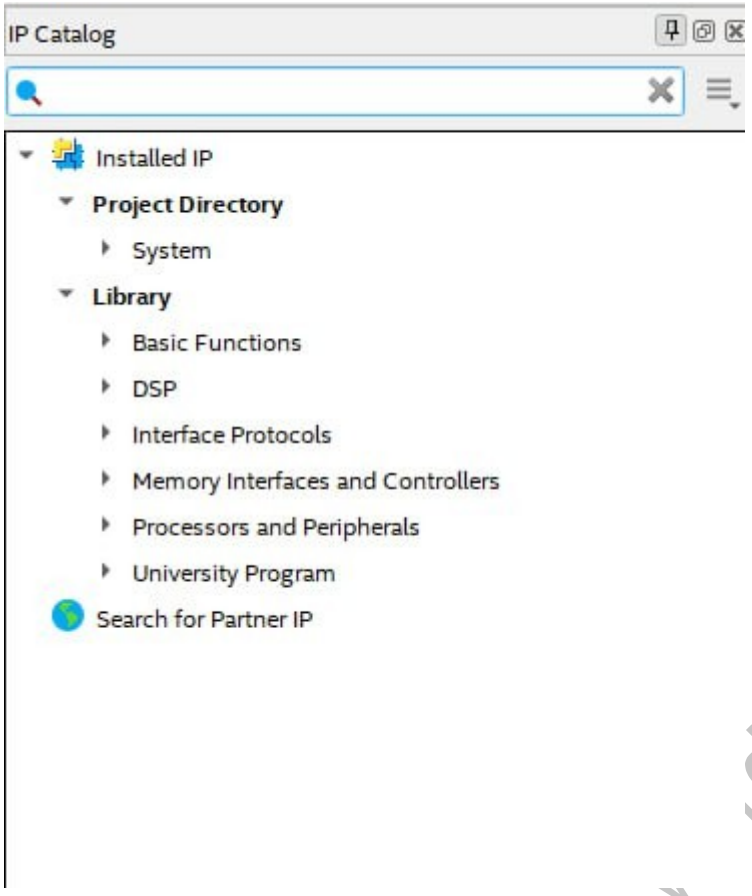
GitHub: https://github.com/DRubioG

Fecha última modificación: 24/02/2025

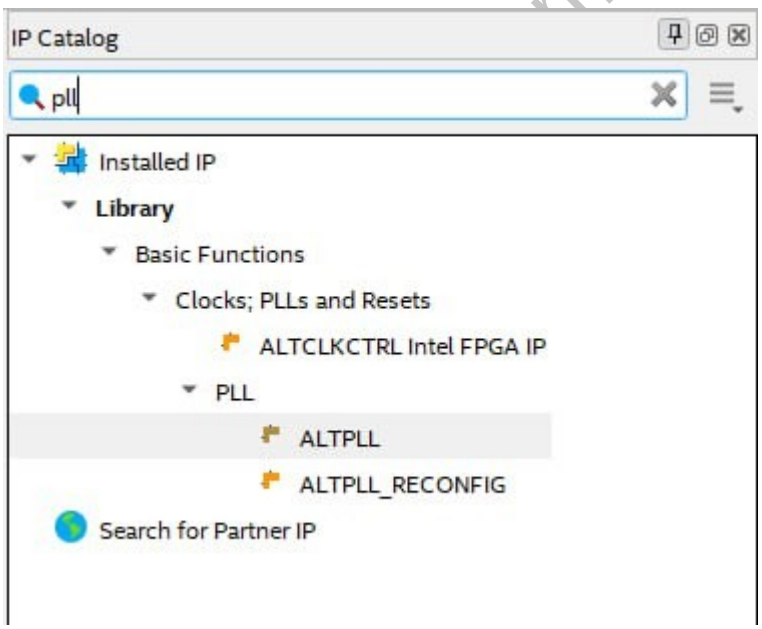
Para añadir un PLL en Quartus, tenemos que ir a *Tools*, y después a *IP Catalog*.



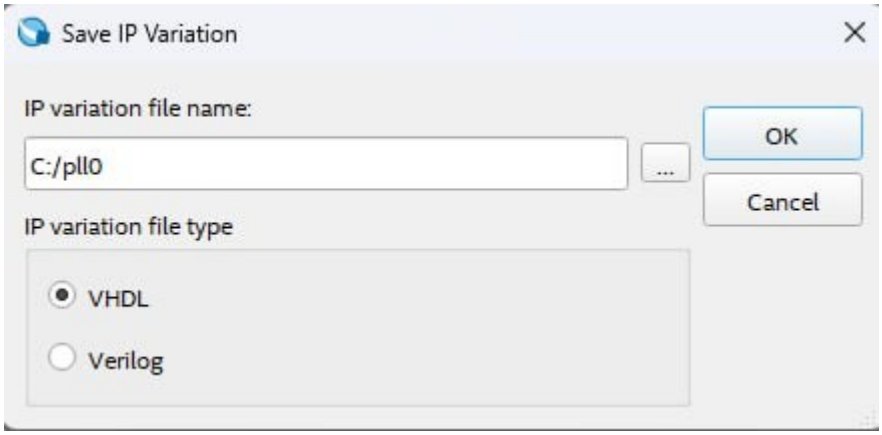
Cuando le demos, a la derecha aparecerá el *IP Catalog*.



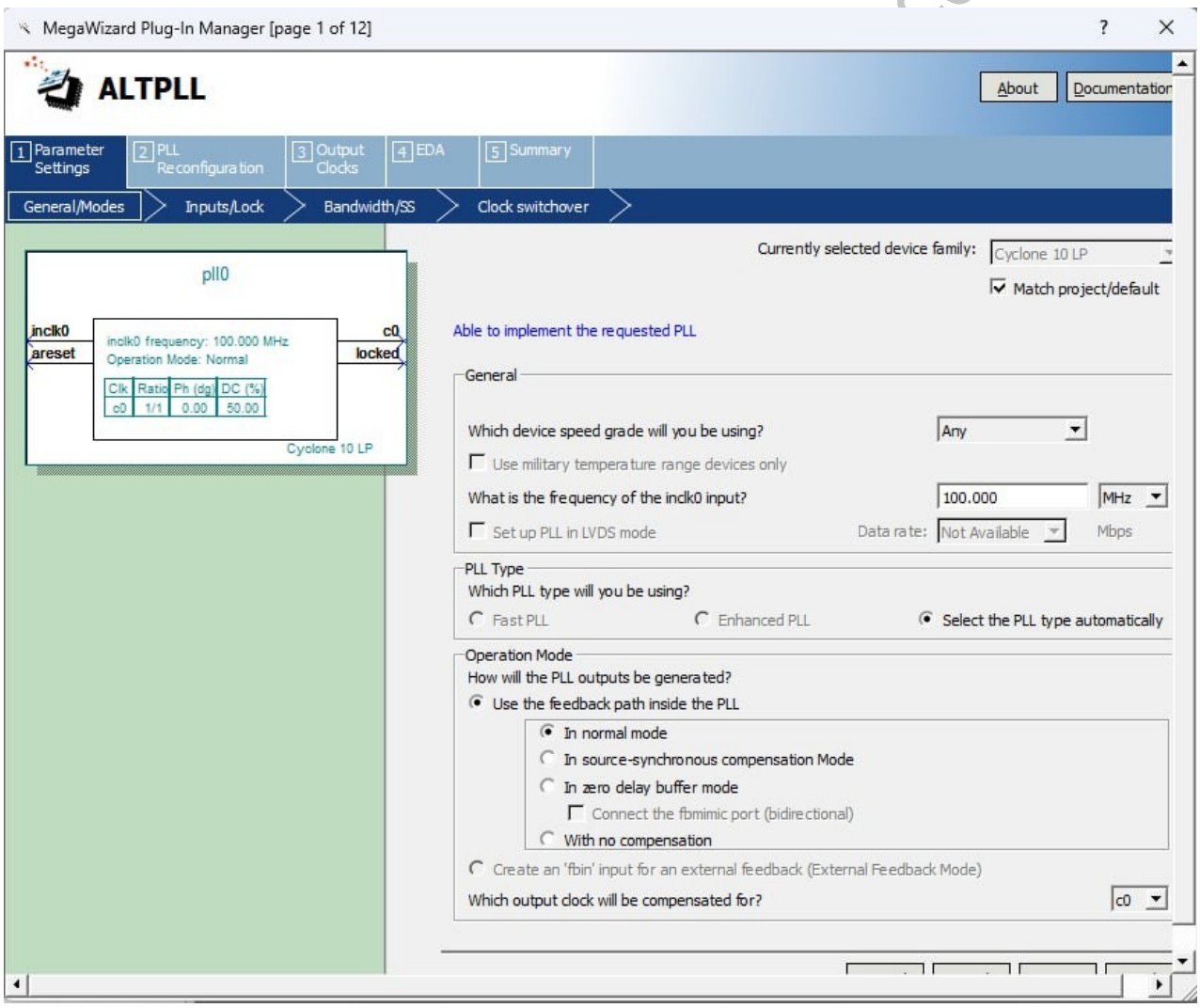
Buscamos el PLL y nos aparece un bloque IP llamado **ALTPLL**.



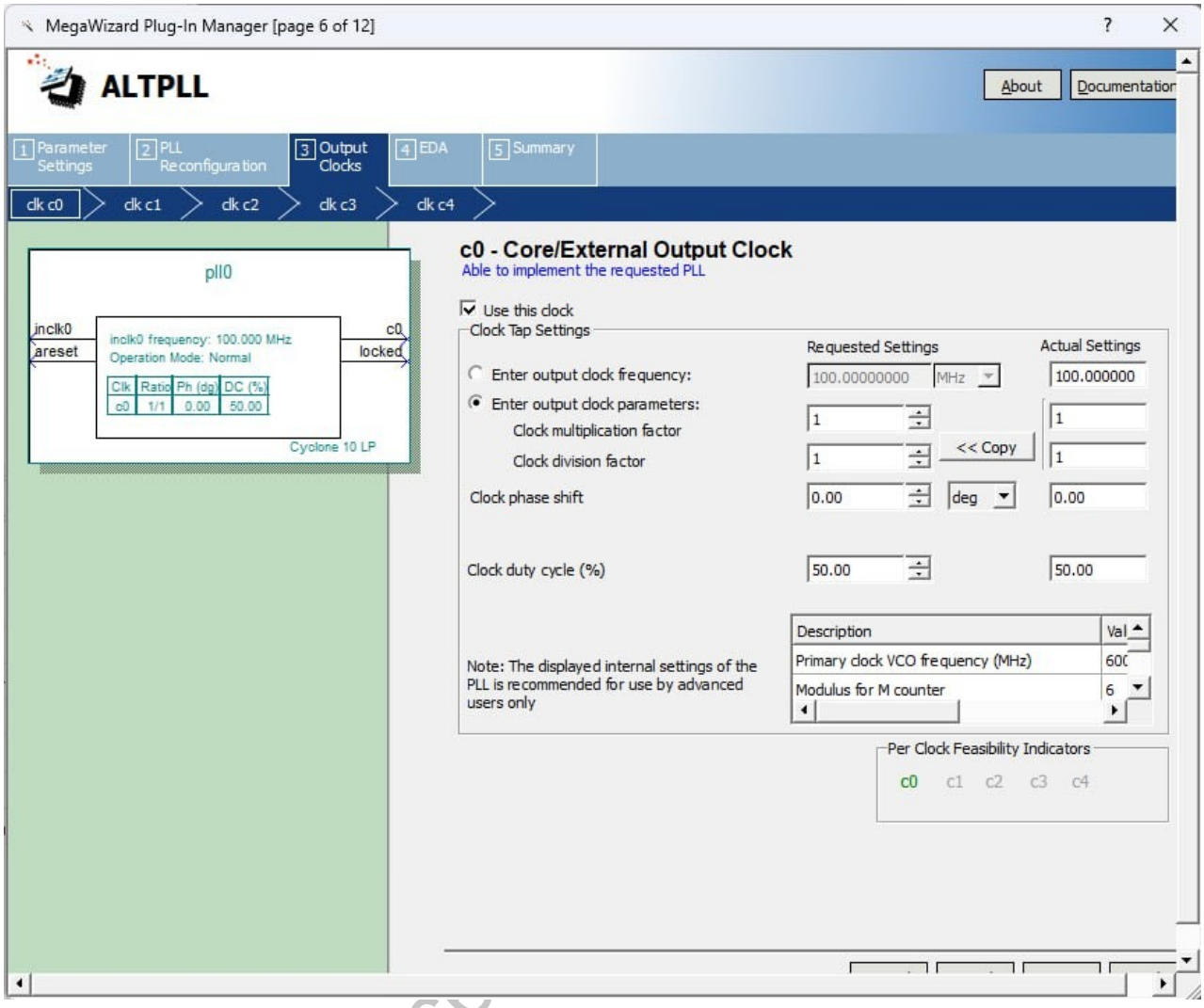
Al darle al bloque IP, lo primero que nos pide es el nombre que le queremos dar y el tipo de fichero en el que se hará la instanciación del bloque IP.



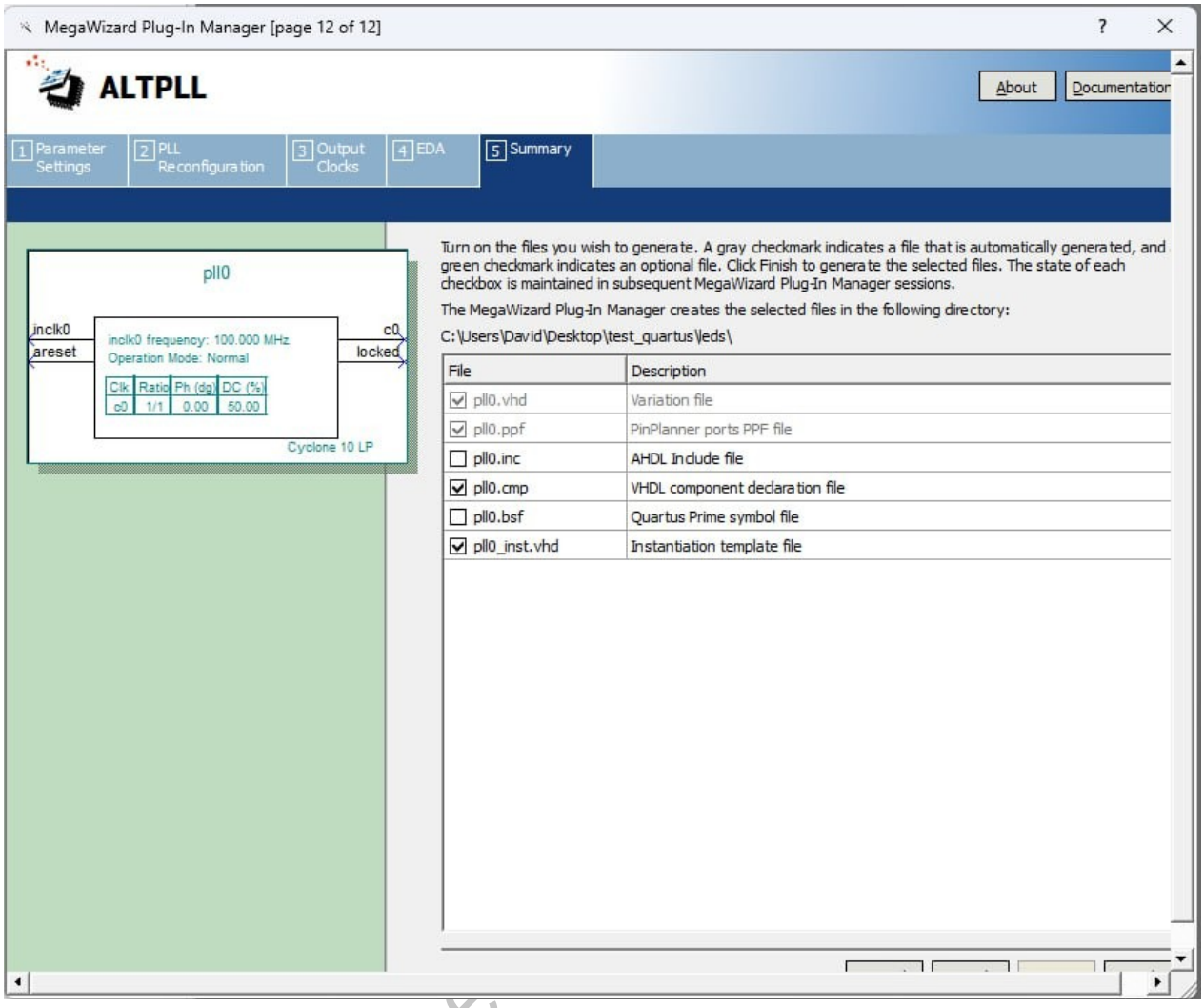
Después, se nos abre la ventana de configuración del PLL donde nos pregunta la frecuencia del reloj de entrada y otros muchos parámetros.



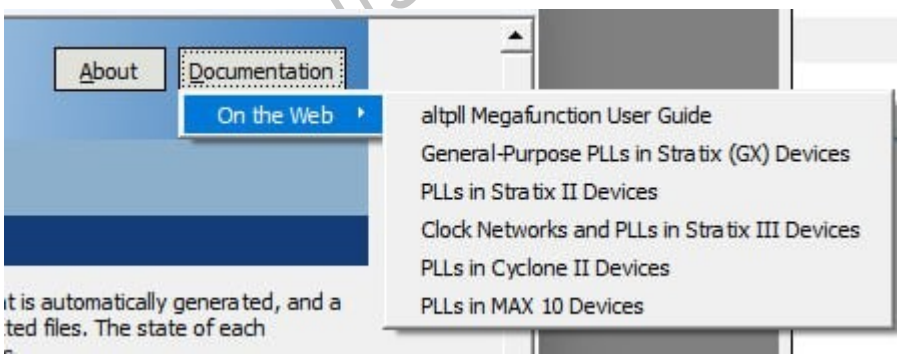
En la opción *Output Clocks* nos permite configurar hasta 5 relojes.



Y en la última pestaña nos hace un resumen de los ficheros que va a crear. Aquí marcamos el último fichero para que nos genere la instanciación.



Si necesitamos más información sobre el bloque IP, solo hace falta darle a *Documentation* y podemos buscar más información.



Una vez hayamos terminado de configurar el PLL nos aparecerá el PLL creado en Files.

Si lo abrimos podemos ver el fichero de instanciación del PLL que da Quartus.

```
ENTITY pll0 IS
    PORT
    (
        areset      : IN STD_LOGIC  := '0';
        inclk0      : IN STD_LOGIC  := '0';
        c0    : OUT STD_LOGIC ;
        locked      : OUT STD_LOGIC
    );
END pll0;

ARCHITECTURE SYN OF pll0 IS

    SIGNAL sub_wire0  : STD_LOGIC ;
    SIGNAL sub_wire1  : STD_LOGIC_VECTOR (1 DOWNTO 0);
    SIGNAL sub_wire2_bv  : BIT_VECTOR (0 DOWNTO 0);
    SIGNAL sub_wire2  : STD_LOGIC_VECTOR (0 DOWNTO 0);
    SIGNAL sub_wire3  : STD_LOGIC_VECTOR (4 DOWNTO 0);
    SIGNAL sub_wire4  : STD_LOGIC ;
    SIGNAL sub_wire5  : STD_LOGIC ;


    COMPONENT altpll
    GENERIC (
        bandwidth_type     : STRING;
        clk0_divide_by     : NATURAL;
        clk0_duty_cycle      : NATURAL;
        clk0_multiply_by      : NATURAL;
        clk0_phase_shift      : STRING;
        compensate_clock      : STRING;
        inclk0_input_frequency      : NATURAL;
        intended_device_family      : STRING;
        lpm_hint     : STRING;
        lpm_type     : STRING;
        operation_mode     : STRING;
        pll_type     : STRING;
        port_activeclock      : STRING;
        port_areset     : STRING;
        port_clkbad0       : STRING;
        port_clkbad1       : STRING;
        port_clkloss       : STRING;
        port_clkswitch     : STRING;
        port_configupdate     : STRING;
        port_fbin       : STRING;
        port_inclk0     : STRING;
        port_inclk1     : STRING;
        port_locked     : STRING;
        port_pfdena     : STRING;
        port_phasecounterselect     : STRING;
        port_phasedone     : STRING;
        port_phasestep     : STRING;
        port_phaseupdown      : STRING;
        port_pllena     : STRING;
        port_scanaclr      : STRING;
        port_scanclk       : STRING;
        port_scanclkena       : STRING;
        port_scandata      : STRING;
        port_scandataout      : STRING;
        port_scandone      : STRING;
        port_scanread      : STRING;
        port_scanwrite     : STRING;
        port_clk0       : STRING;
        port_clk1       : STRING;
        port_clk2       : STRING;
        port_clk3       : STRING;
        port_clk4       : STRING;
        port_clk5       : STRING;
        port_clkena0       : STRING;
```


Ahora lo único que tenemos que hacer es instanciar el bloque en nuestro código.

```
component pll0 IS
    PORT
    (
        areset      : IN STD_LOGIC  := '0';
        inclk0      : IN STD_LOGIC  := '0';
        c0    : OUT STD_LOGIC ;
        locked      : OUT STD_LOGIC
    );
END component;

begin

inst_pll : pll0
    PORT map
    (
        areset      => rst_n,
        inclk0      => clk,
        c0    => clk_aux,
        locked      => open
    );
```

Si ahora sintetizamos y vemos el resultado, podemos ver que aparece un bloque en la síntesis, que se corresponde con el PLL que hemos instanciado.