

Cómo añadir un PLL en Quartus

Creador: David Rubio G.

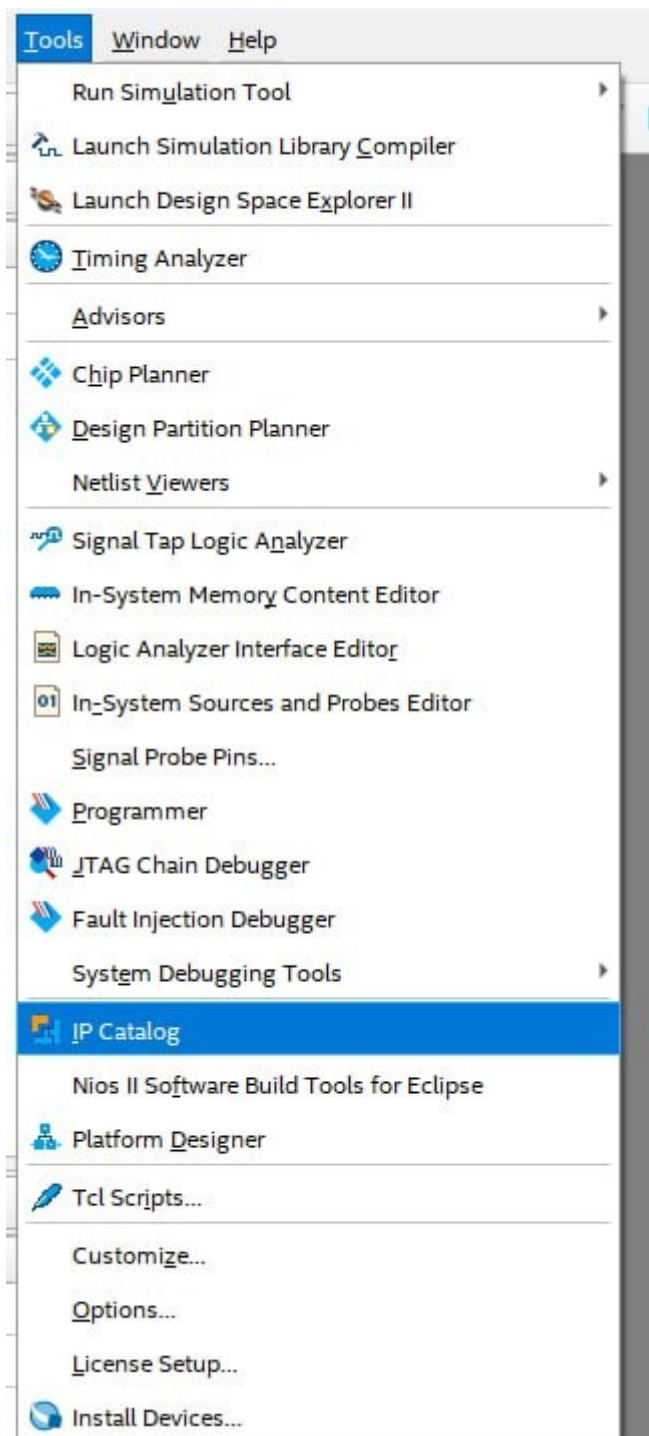
Entrada: <https://soceame.wordpress.com/2025/01/14/como-anadir-un-pll-en-quartus/>

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

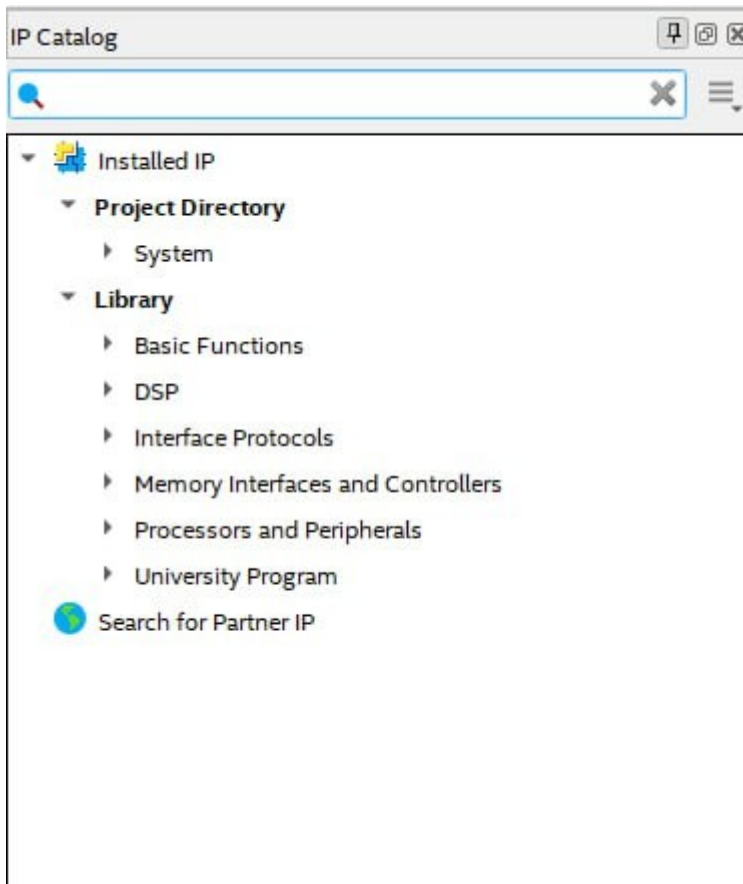
GitHub: <https://github.com/DRubioG>

Fecha última modificación: 24/02/2025

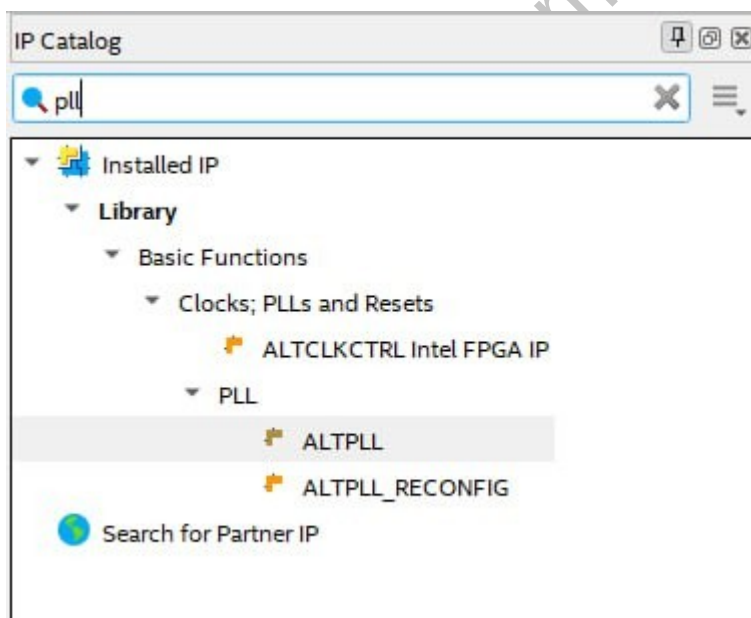
Para añadir un PLL en Quartus, tenemos que ir a *Tools*, y después a *IP Catalog*.



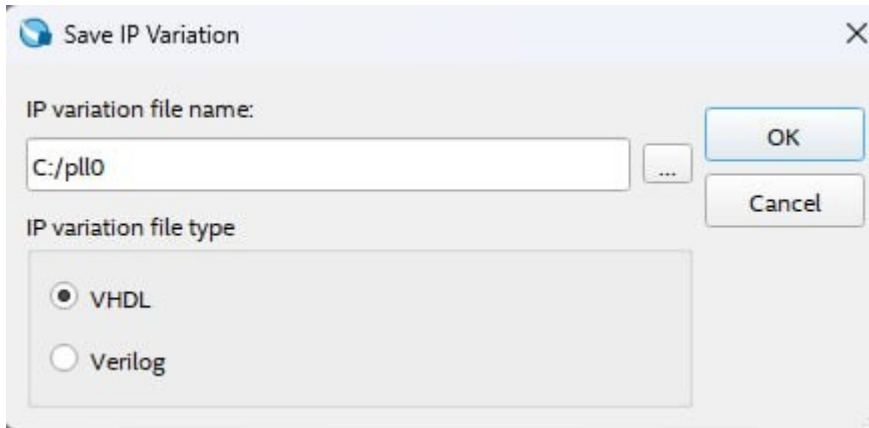
Cuando le demos, a la derecha aparecerá el *IP Catalog*.



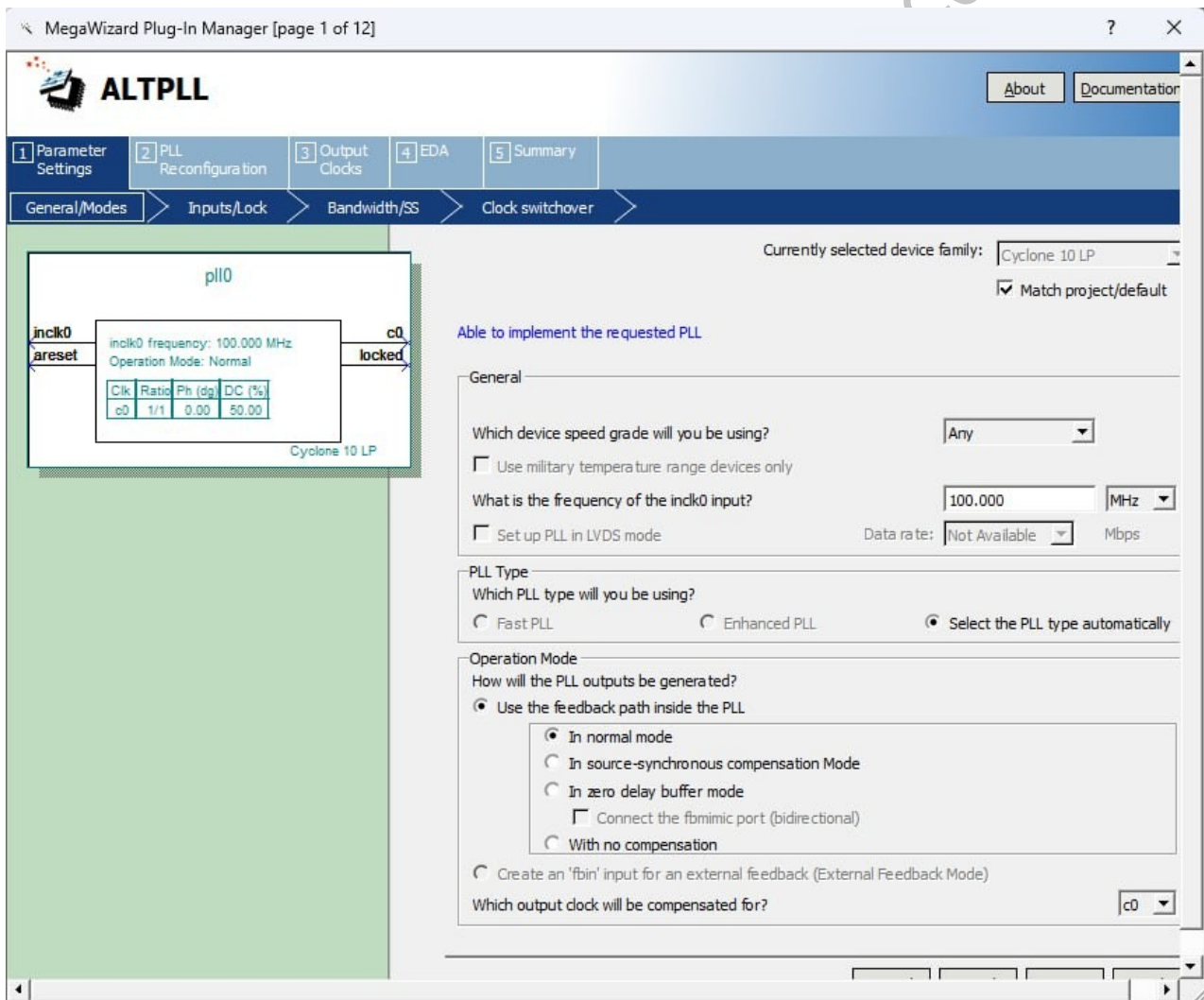
Buscamos el PLL y nos aparece un bloque IP llamado **ALTPLL**.



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.



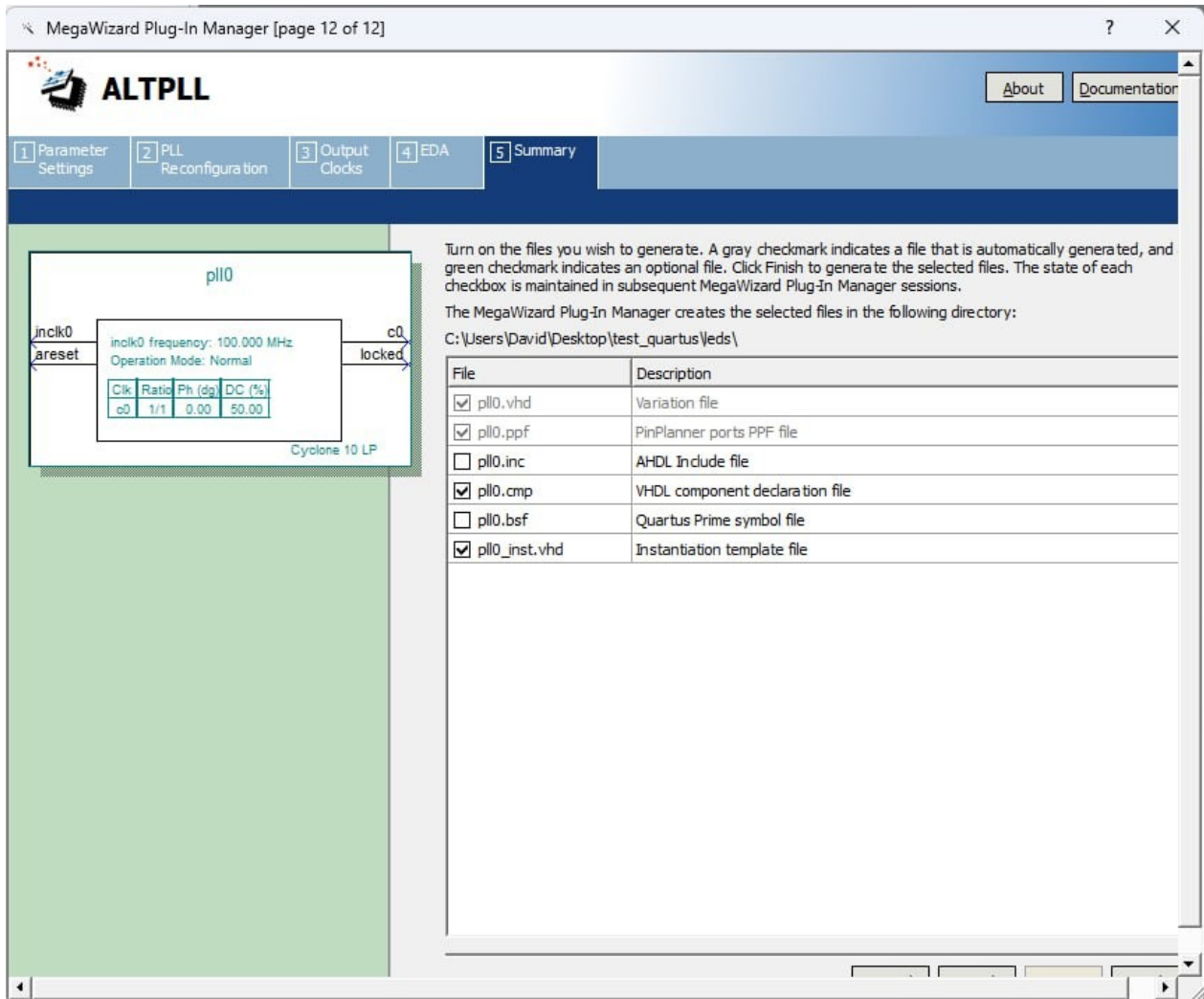
En la opción *Output Clocks* nos permite configurar hasta 5 relojes.

The screenshot shows the MegaWizard Plug-In Manager (ALTPLL) window. The title bar indicates it is page 6 of 12. The window has a top navigation bar with tabs: 1 Parameter Settings, 2 PLL Reconfiguration, 3 Output Clocks (selected), 4 EDA, and 5 Summary. Below the tabs is a breadcrumb trail: clk c0 > clk c1 > clk c2 > clk c3 > clk c4. The main area is titled 'c0 - Core/External Output Clock' with the subtitle 'Able to implement the requested PLL'. On the left, there is a block diagram of the PLL. It shows an input 'inclk0' and a reset signal 'areset' entering a block labeled 'pll0'. Inside the block, it says 'inclk0 frequency: 100.000 MHz' and 'Operation Mode: Normal'. Below this is a table with columns 'Clk', 'Ratio', 'Ph (deg)', and 'DC (%)'. The row for 'c0' shows a ratio of '1/1', phase of '0.00', and duty cycle of '50.00'. The output of the block is 'c0', which is labeled 'locked'. The block is identified as 'Cyclone 10 LP'. On the right, there are 'Clock Tap Settings'. A checkbox 'Use this clock' is checked. There are two options: 'Enter output clock frequency' (radio button) and 'Enter output clock parameters' (radio button, selected). The 'Enter output clock parameters' section has fields for 'Clock multiplication factor' (1), 'Clock division factor' (1), 'Clock phase shift' (0.00 deg), and 'Clock duty cycle (%)' (50.00). To the right of these fields are 'Requested Settings' and 'Actual Settings' columns. The 'Requested Settings' column has a value of 100.0000000 MHz. The 'Actual Settings' column has a value of 100.000000. Below these fields is a '<< Copy' button. At the bottom right, there is a 'Per Clock Feasibility Indicators' section with a row of indicators: c0 (green), c1, c2, c3, and c4. A note at the bottom left states: 'Note: The displayed internal settings of the PLL is recommended for use by advanced users only'. Below the note is a table with columns 'Description' and 'Val'. The table has two rows: 'Primary clock VCO frequency (MHz)' with a value of 600, and 'Modulus for M counter' with a value of 6.

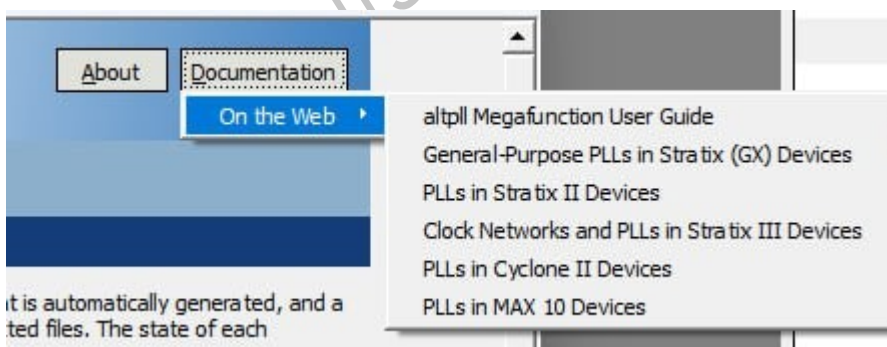
Requested Settings	Actual Settings
100.0000000 MHz	100.000000
1	1
1	1
0.00 deg	0.00
50.00	50.00

Description	Val
Primary clock VCO frequency (MHz)	600
Modulus for M counter	6

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 altp11
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_scanclock : 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_clkkena0 : 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.

