Implementar pines diferenciales en Quartus

Creador: David Rubio G.

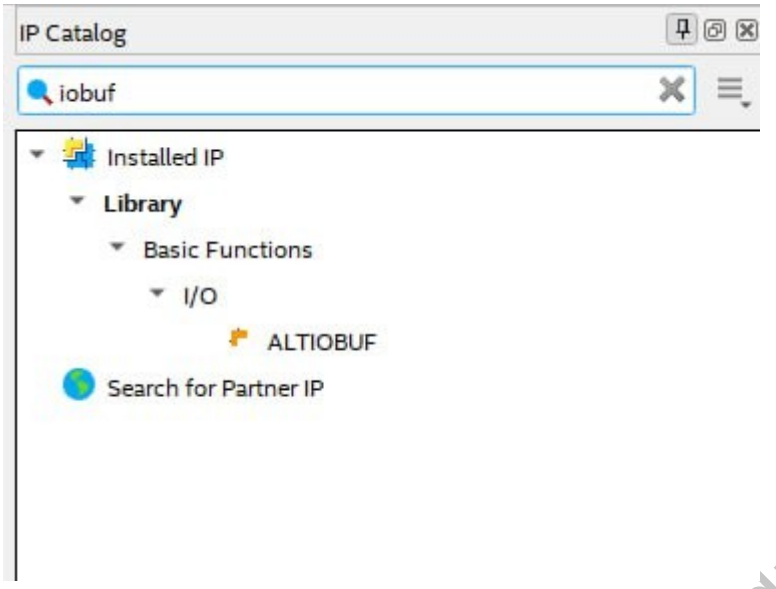
Entrada: https://soceame.wordpress.com/2025/01/14/implementar-pines-diferenciales-en-quartus/

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

GitHub: https://github.com/DRubioG

Fecha última modificación: 24/02/2025

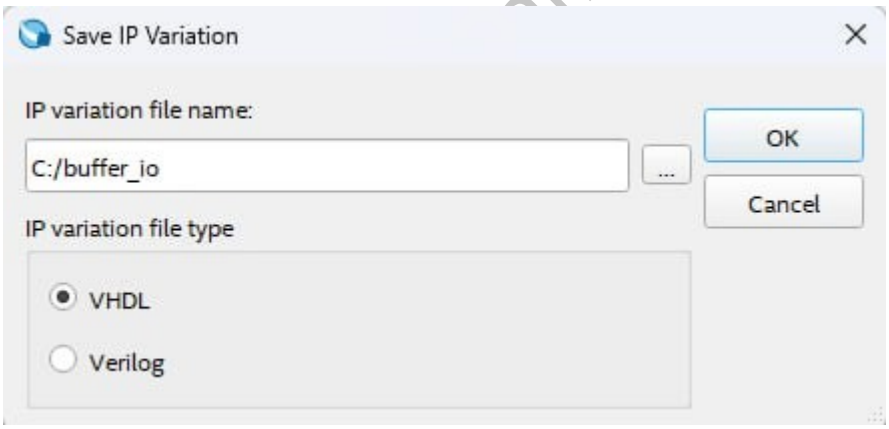
Para acceder a los buffer en Quartus, primero se accede al *IP Catalog*, y en él se busca el bloque IP **ALTIOBUF**.



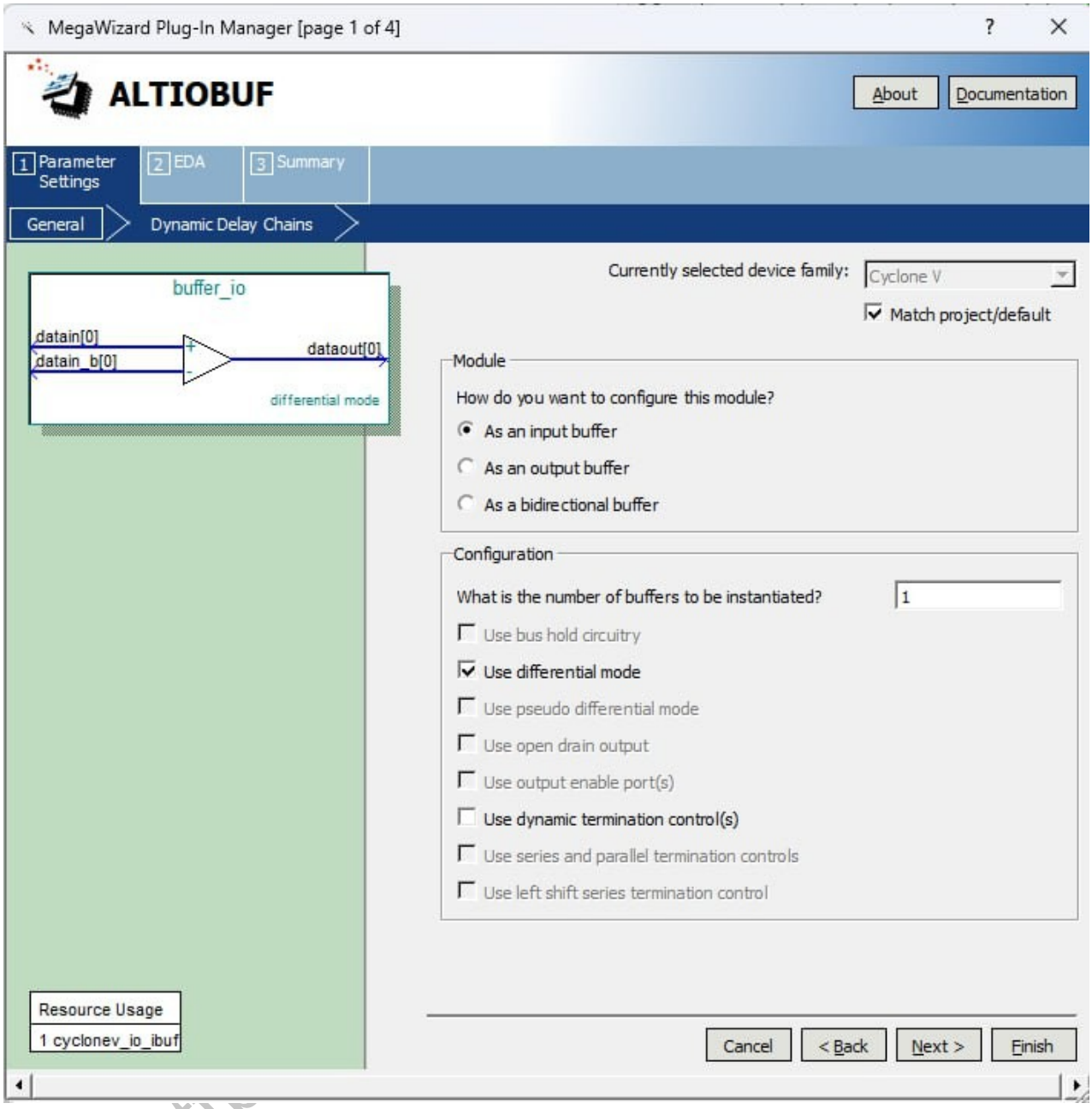
NOTA: los diferentes buffers que se pueden instanciar con el bloque *ALTIOBUF* están descritos en la siguiente entrada.

https://soceame.wordpress.com/2025/01/14/buffers-en-quartus/

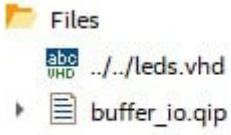
Al intentar instanciarlo lo primero que nos pide



Después se abre el editor, en él configuramos el buffer como entrada y marcamos la casilla diferencial.



Cuando terminemos, nos genera un .qip (es posible que para exportar el bloque IP tengamos que haber marcado la casilla para que nos genere un modelo de instanciación).



Si miramos el fichero que hay dentro, podemos ver la declaración del bloque IP. Solo tenemos que instanciar este fichero en FW.

```
LIBRARY cyclonev;
USE cyclonev.all;

--synthesis_resources = cyclonev_io_ibuf 1
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY  buffer_io_iobuf_in_6pi IS
    PORT
    (
       datain  :  IN  STD_LOGIC_VECTOR (0 DOWNTO 0);
       datain_b   :  IN  STD_LOGIC_VECTOR (0 DOWNTO 0) := (OTHERS => '0');
       dataout :  OUT  STD_LOGIC_VECTOR (0 DOWNTO 0)
    );
END buffer_io_iobuf_in_6pi;

ARCHITECTURE RTL OF buffer_io_iobuf_in_6pi IS

    SIGNAL  wire_ibufa_o   :  STD_LOGIC;
    COMPONENT  cyclonev_io_ibuf
    GENERIC
    (
      bus_hold :  STRING := "false";
      differential_mode :  STRING := "false";
      simulate_z_as  :  STRING := "z";
      lpm_type :  STRING := "cyclonev_io_ibuf"
    );
    PORT
```

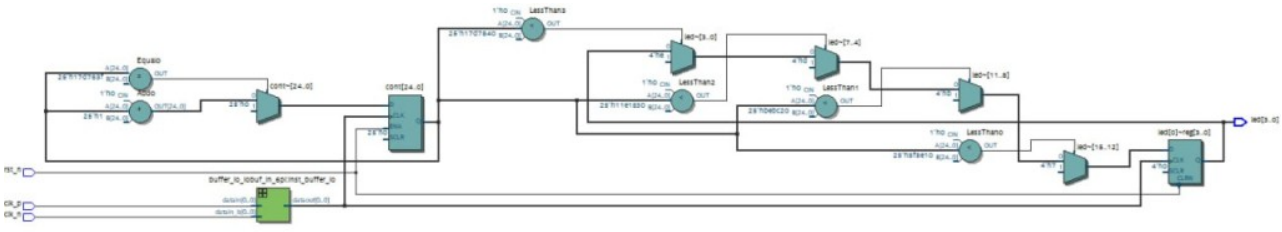
Ahora lo que tenemos que hacer es instanciar el bloque IP en nuestro FW.

```
component  buffer_io_iobuf_in_6pi IS
    PORT
    (
       datain  :  IN  STD_LOGIC;
       datain_b   :  IN  STD_LOGIC;
       dataout :  OUT  STD_LOGIC
    );
 END component;

begin

inst_buffer_io : buffer_io_iobuf_in_6pi
    PORT map
    (
       datain  => clk_p,
       datain_b   => clk_n,
       dataout => clk
    );
```

Y ahora sintetizamos.



Y si lo miramos bien podemos ver el bloque que se encarga de la señal diferencial.