Instantiating IP blocks in FW in Libero without using a SmartDesign

Created by: David Rubio G.

Blog post: https://soceame.wordpress.com/2025/03/11/instantiating-ip-blocks-in-fw-in-libero-without-using-a-smartdesign/

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

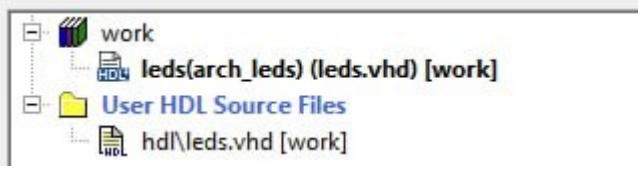
GitHub: https://github.com/DRubioG

Last modification date: 11/03/25

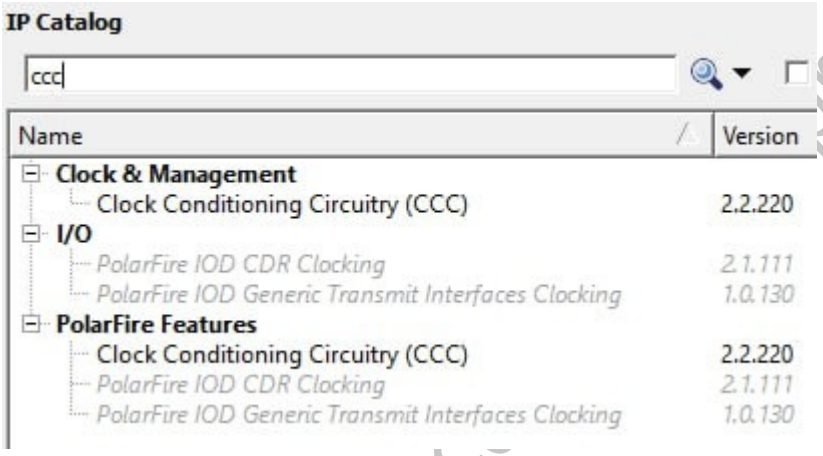
In this post I will explain how to instantiate an IP block in FW in Libero. To do this, a real example will be used, for which a PLL is to be placed in a block of LEDs created by the user.

Development

The first thing you need is the FW module in which you are going to instantiate the IP block.

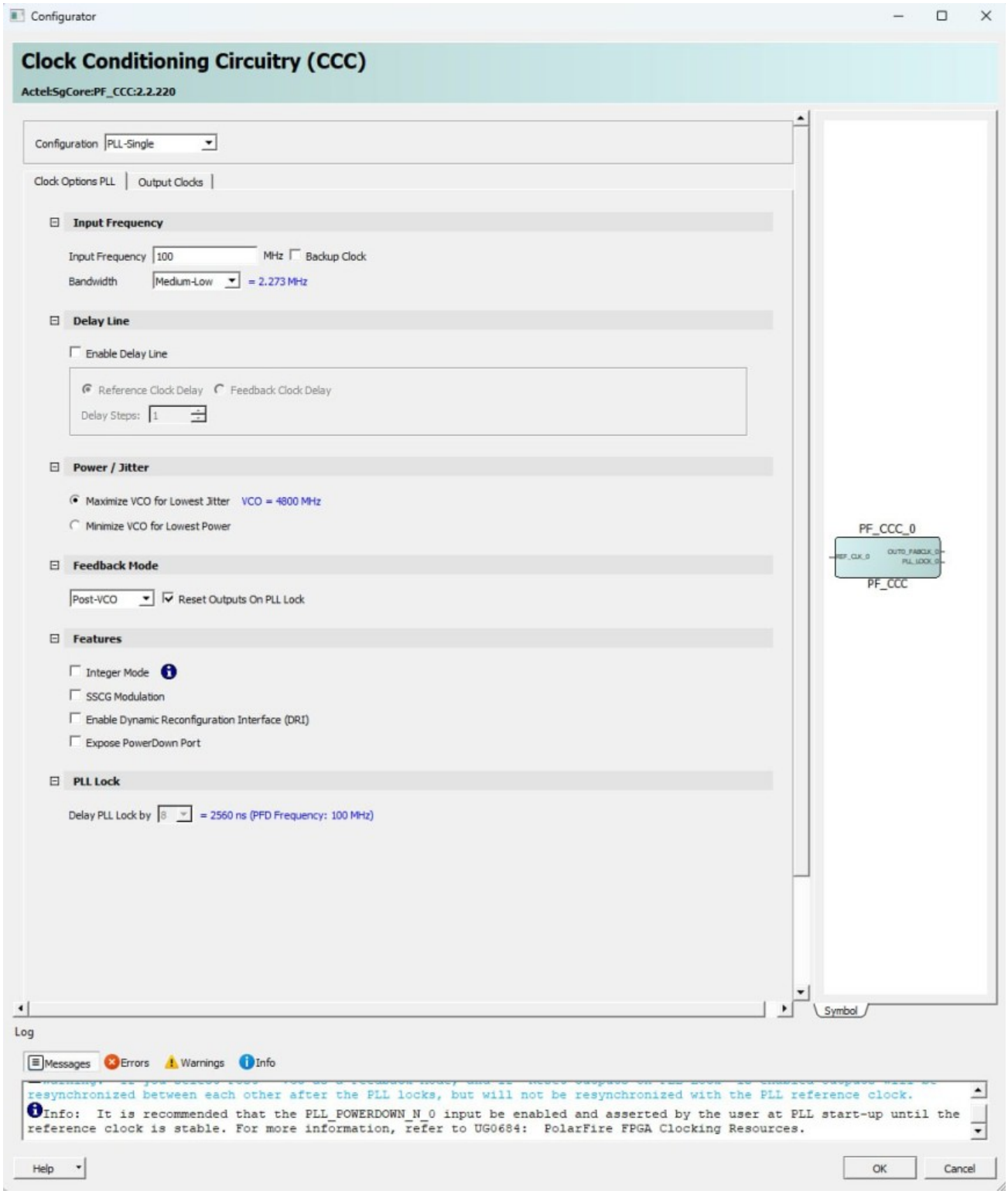


Now you go to the Catalog, here you choose the IP block you want to instantiate. In my case I am going to instantiate a PLL.

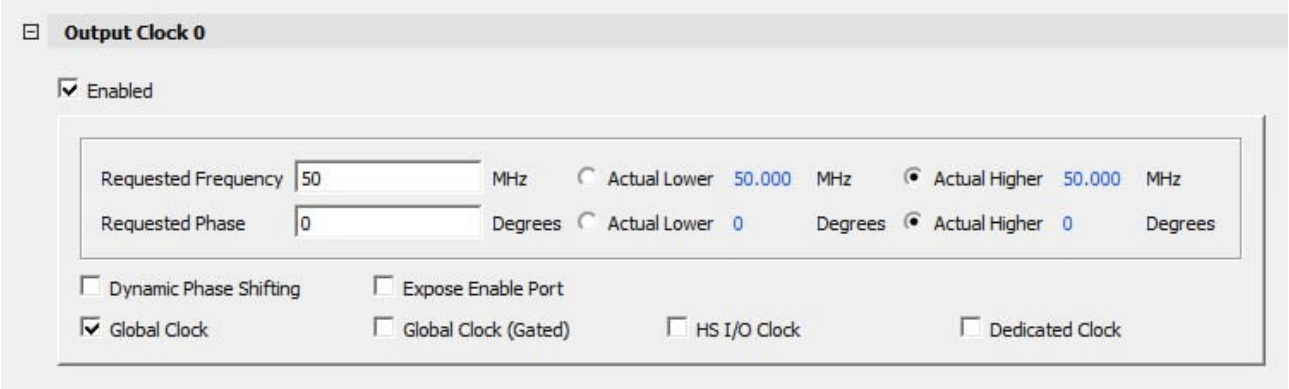


When you click on it, the PLL configuration tab opens. Here we configure the PLL as we want.

https://soceame.wordpress.com/2025/03/11/instantiating-ip-blocks-in-fw-in-libero-without-using-a-smartdesign/

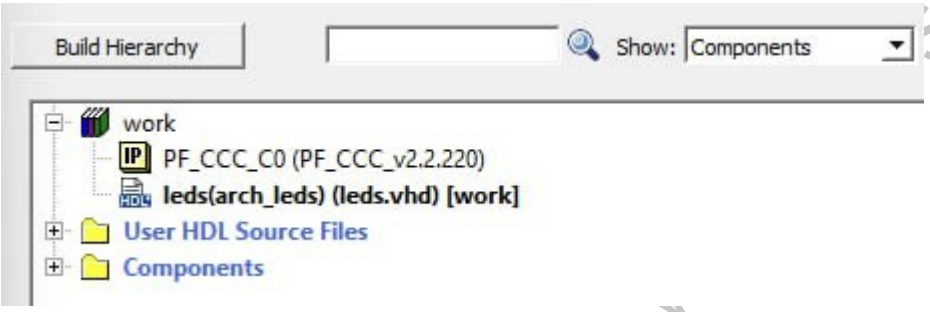


We configure it with the desired output frequency.

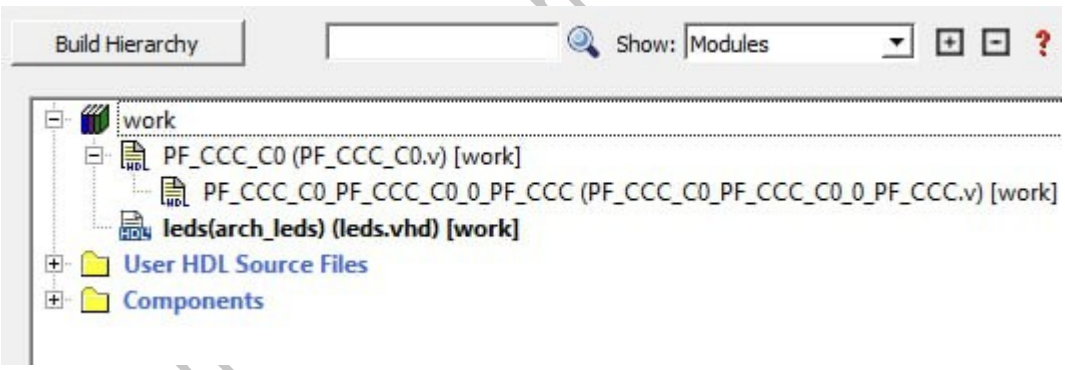


When you finish the configuration, the IP block appears in the hierarchy at the same level as the FW.

NOTE: *this would be where you normally drag the IP block to a SmartDesign and join it.*



Now what you have to do is change the view from the *Components* hierarchy to the *Modules* view.



Now you can see two modules to implement the PLL, both in Verilog. We are only interested in the top module of the PLL, the *PF_CCC_C0*.

If we open it we can see that the port system is the same as the PLL we have configured.

```
// PF_CCC_C0
module PF_CCC_C0(
    // Inputs
    REF_CLK_0,
    // Outputs
    OUT0_FABCLK_0,
    PLL_LOCK_0
);

//--------------------------------------------------------------------
// Input
//--------------------------------------------------------------------
input  REF_CLK_0;
//--------------------------------------------------------------------
// Output
//--------------------------------------------------------------------
output OUT0_FABCLK_0;
output PLL_LOCK_0;
//--------------------------------------------------------------------
// Nets
//--------------------------------------------------------------------
wire   OUT0_FABCLK_0_net_0;
wire   PLL_LOCK_0_net_0;
wire   REF_CLK_0;
wire   OUT0_FABCLK_0_net_1;
wire   PLL_LOCK_0_net_1;
//--------------------------------------------------------------------
// TiedOff Nets
//--------------------------------------------------------------------
wire   GND_net;
wire   [10:0]DRI_CTRL_0_const_net_0;
wire   [32:0]DRI_WDATA_0_const_net_0;
wire   [10:0]DRI_CTRL_1_const_net_0;
wire   [32:0]DRI_WDATA_1_const_net_0;
```

Now all we need to do is take the ports from the Verilog file.

```
// PF_CCC_C0
module PF_CCC_C0(
    // Inputs
    REF_CLK_0,
    // Outputs
    OUT0_FABCLK_0,
    PLL_LOCK_0
);
```

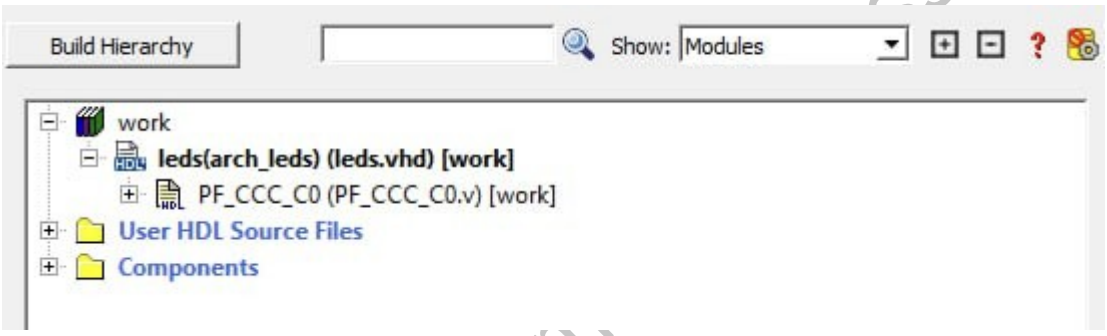
Now we paste these ports into our project. And if we are working in VHDL we convert it into a *component* with the information given to us by the file.

```
component PF_CCC_C0
port (
     REF_CLK_0 : in std_logic;
     OUT0_FABCLK_0 : out std_logic;
     PLL_LOCK_0 : out std_logic
);
end component;
```

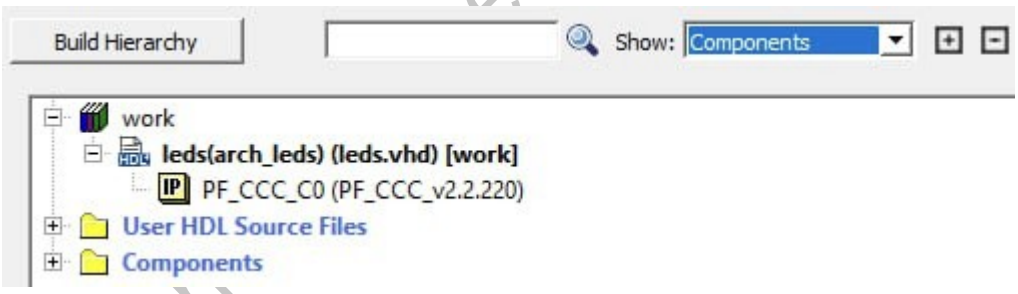
Now we only have to instantiate the IP block with our FW.

```
CC_inst : PF_CCC_C0
port map (
     REF_CLK_0 => clk,
     OUT0_FABCLK_0 => clk_i,
     PLL_LOCK_0 => open
);
```

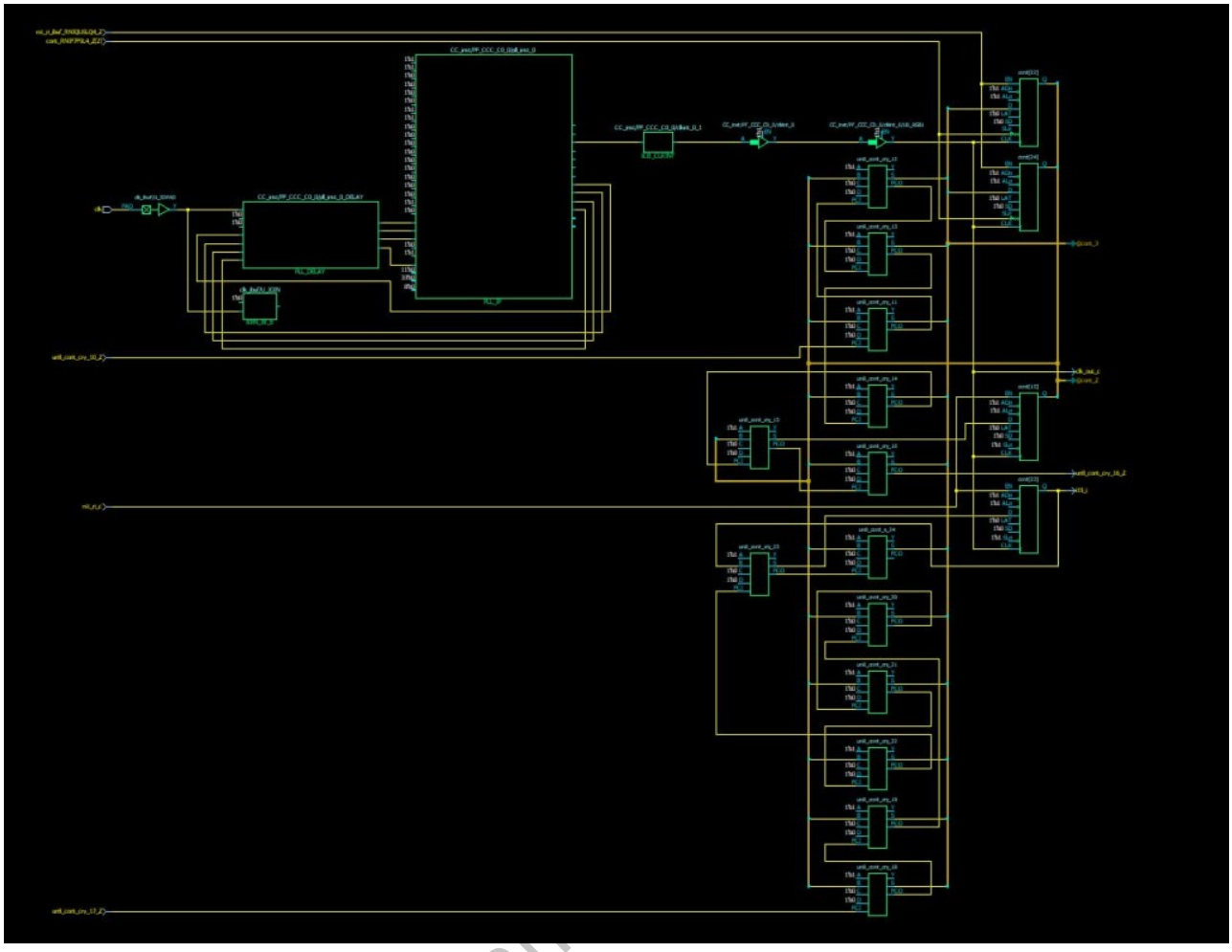
By doing so the hierarchy shows us that the PLL is downstream of the FW module.



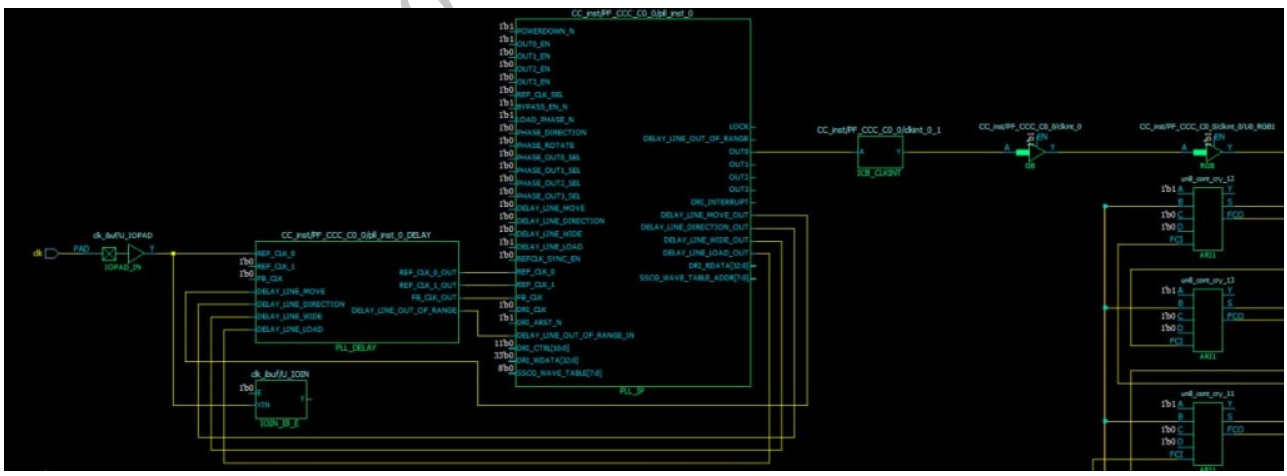
If we change the view from *Modules to Components* the structure is better reflected.



If we now synthesize the FW, we can see that Libero does not give any kind of error, and that it synthesizes the entire FW, including the PLL.



And if we look closely at the synthesis we can see that there is a part that corresponds directly to the PLL we have instantiated.



FINAL NOTE

Well, the same thing that has been explained with a PLL can be done with all the Libero IP blocks. And thus, you don’t have to use SmartDesign.