University of California, Davis

Department of Electrical and Computer Engineering

Tutorial: Instantiating and Using a PLL on the DE10-LITE

Objective: This tutorial explains how to configure and instantiate a Phase-Locked Loop (PLL) for the MAX10 FPGA in Quartus.

Introduction

A Phase-Locked Loop (PLL) is a closed-loop frequency control circuit that compares the phase difference between an input signal and output signal of a voltage-controlled oscillator. The negative feedback loop forces the PLL's output signal to be phase locked with the input signals.

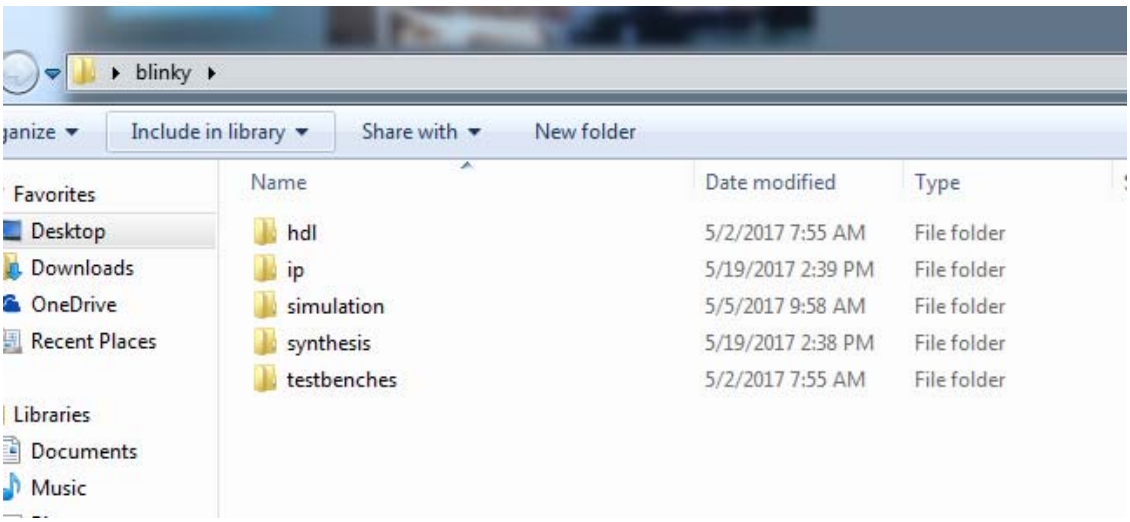
Almost all FPGA's contain some kind of PLL on-chip for clock generation and distribution. One of the big advantages of PLLs is their ability to generate an output clock at a different frequency from the reference input clock. For example, it is entirely possible to generate a 100 MHz internal clock on the MAX10 FPGAs from the 50 MHz external clock on the DE10-LITE board using a PLL.

This tutorial will demonstrate how to use the IP (intellectual property) catalog in Quartus to instantiate a PLL in your design to generate different clock frequencies. A more in depth discussion on PLLs and the Altera/Intel design component may be found at

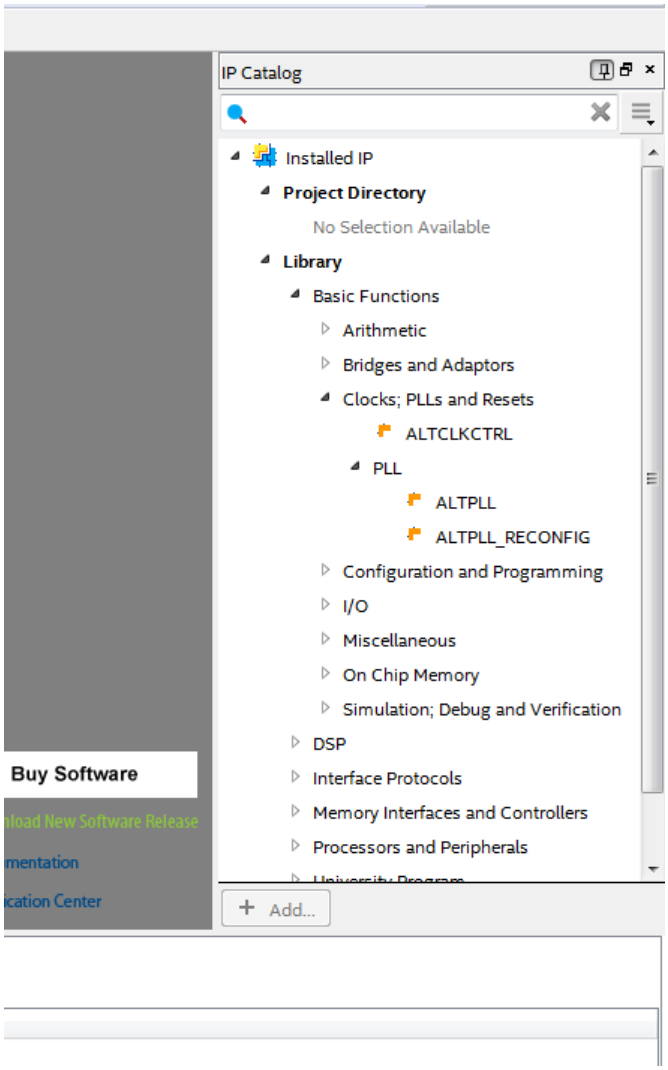
https://www.altera.com/en_US/pdfs/literature/ug/ug_altpll.pdf

Generating the PLL IP component

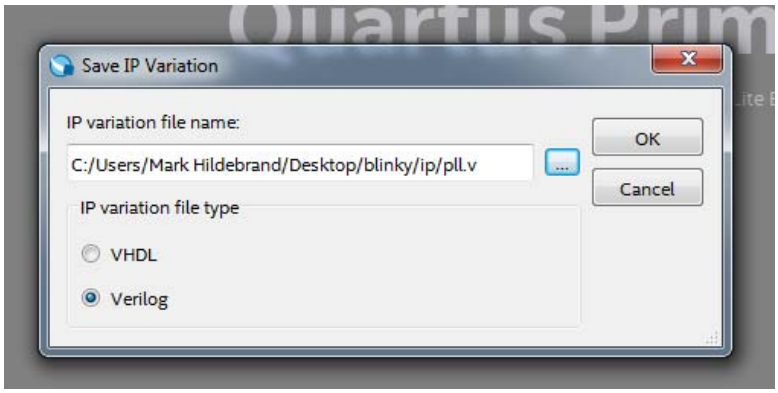
1. Launch Quartus and open the project where you want the PLL. The demonstration project for this tutorial will be called "blinky". A image of the directory structure for "blinky" is shown below:



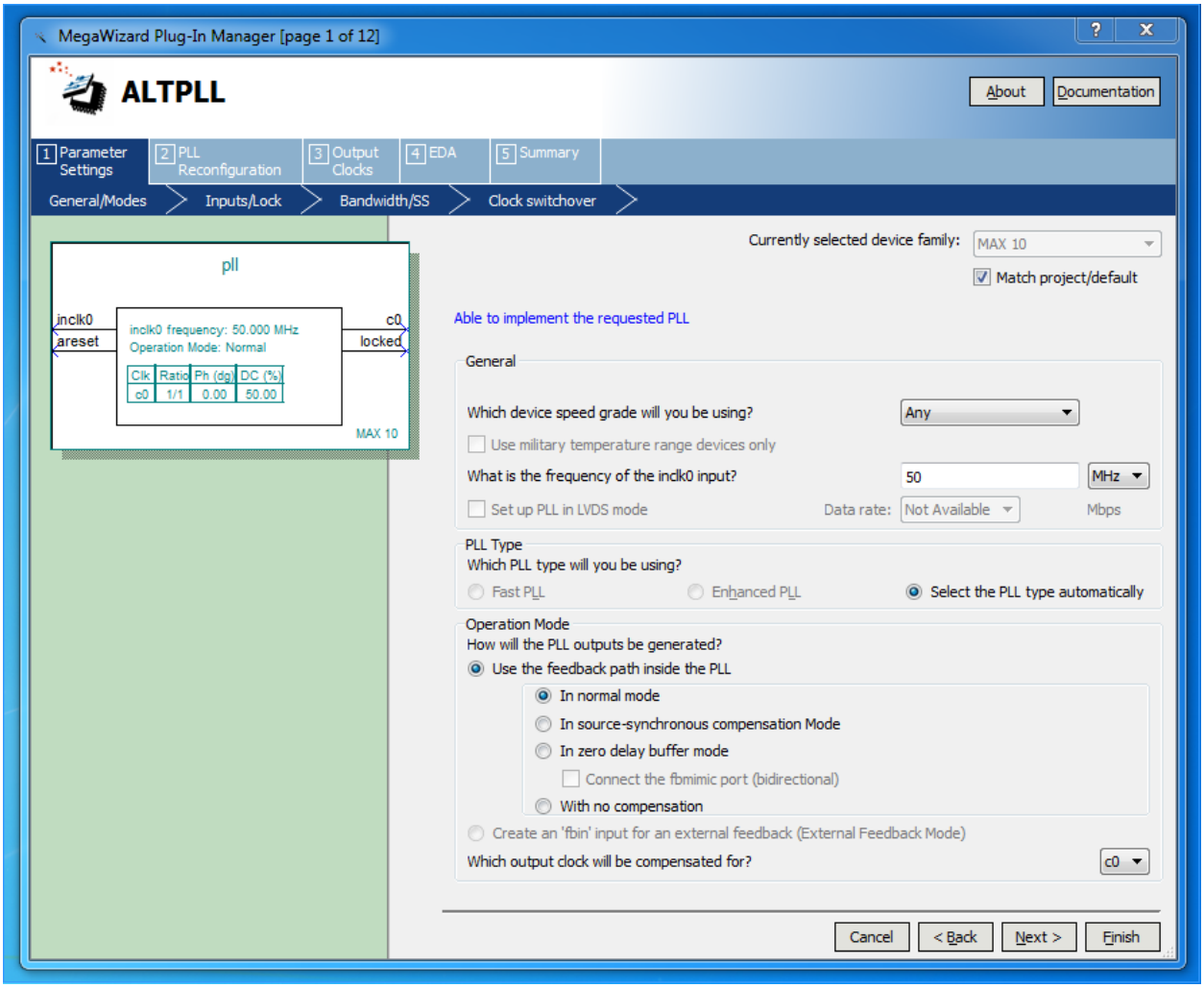
2. In Quartus, open the IP catalog window by clicking **Tools > IP Catalog**. The Catalog window should open in Quartus. Navigate through the library until you find the **ALTPLL** component



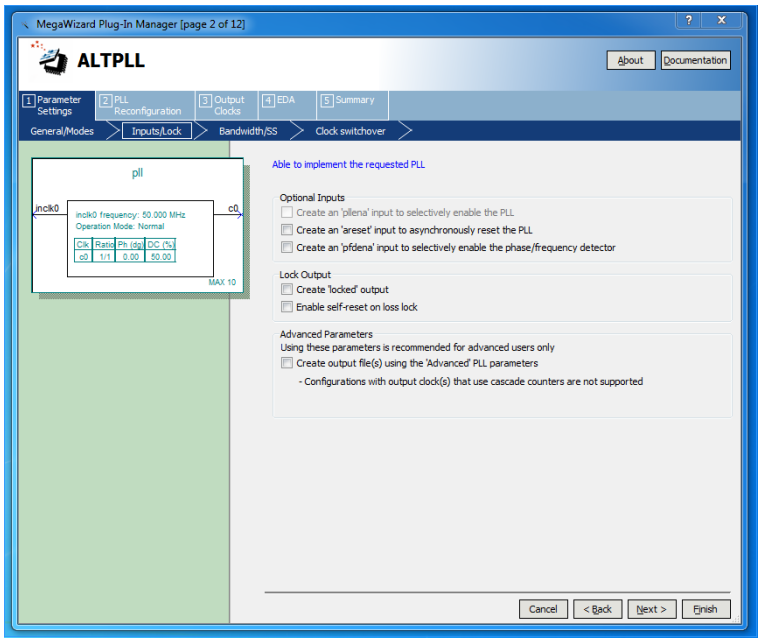
3. Double click on **ALTPLL**. A prompt should appear asking where you want to save your IP component. I usually have a folder for my IP components, so that is where I save my design. Save it where ever is reasonable for you. In this tutorial, the component will be uncreatively called **pll.**



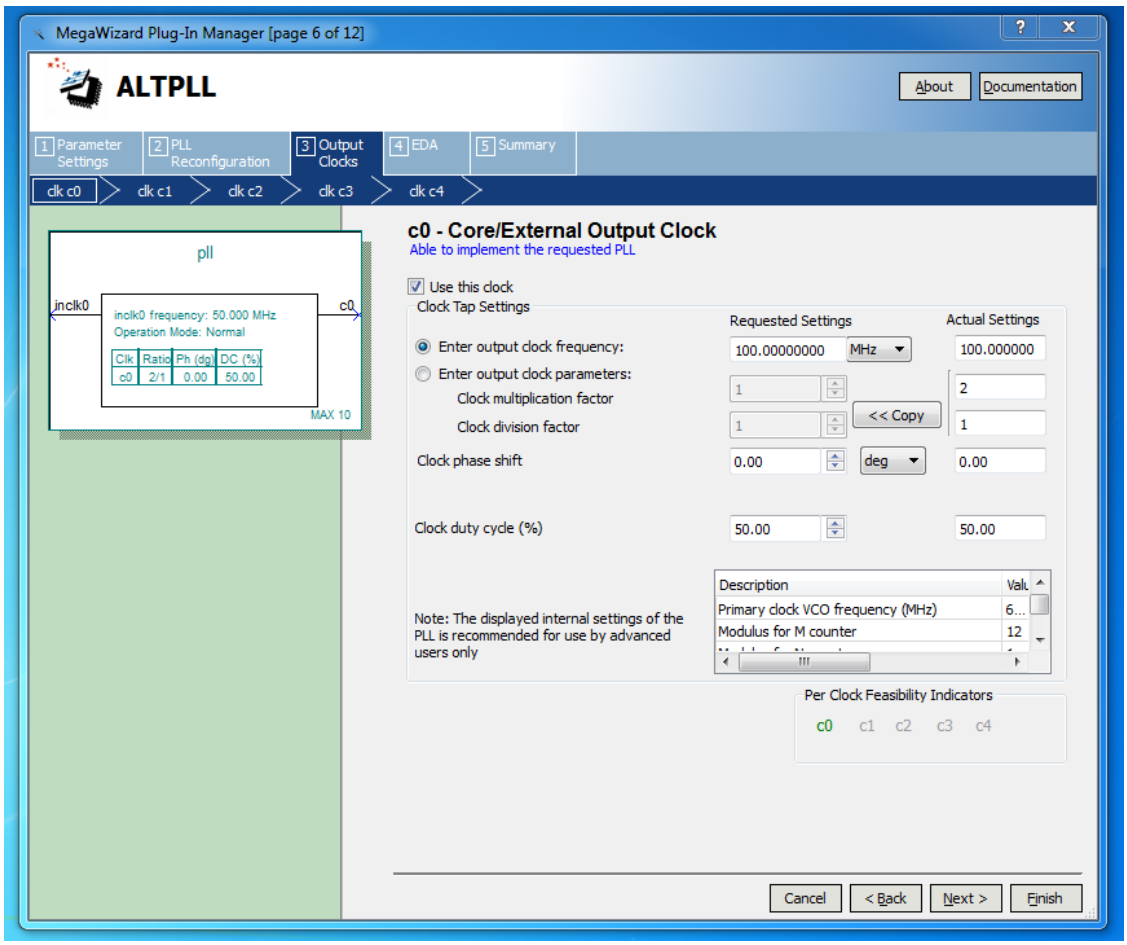
4. Press **OK**. The MegaFunction Plug-In Manager should appear. Make sure you change the **inclk0 input frequency** to 50 MHz to match the speed of the external clock on the DE10-LITE board. This is important for the PLL to function correctly.



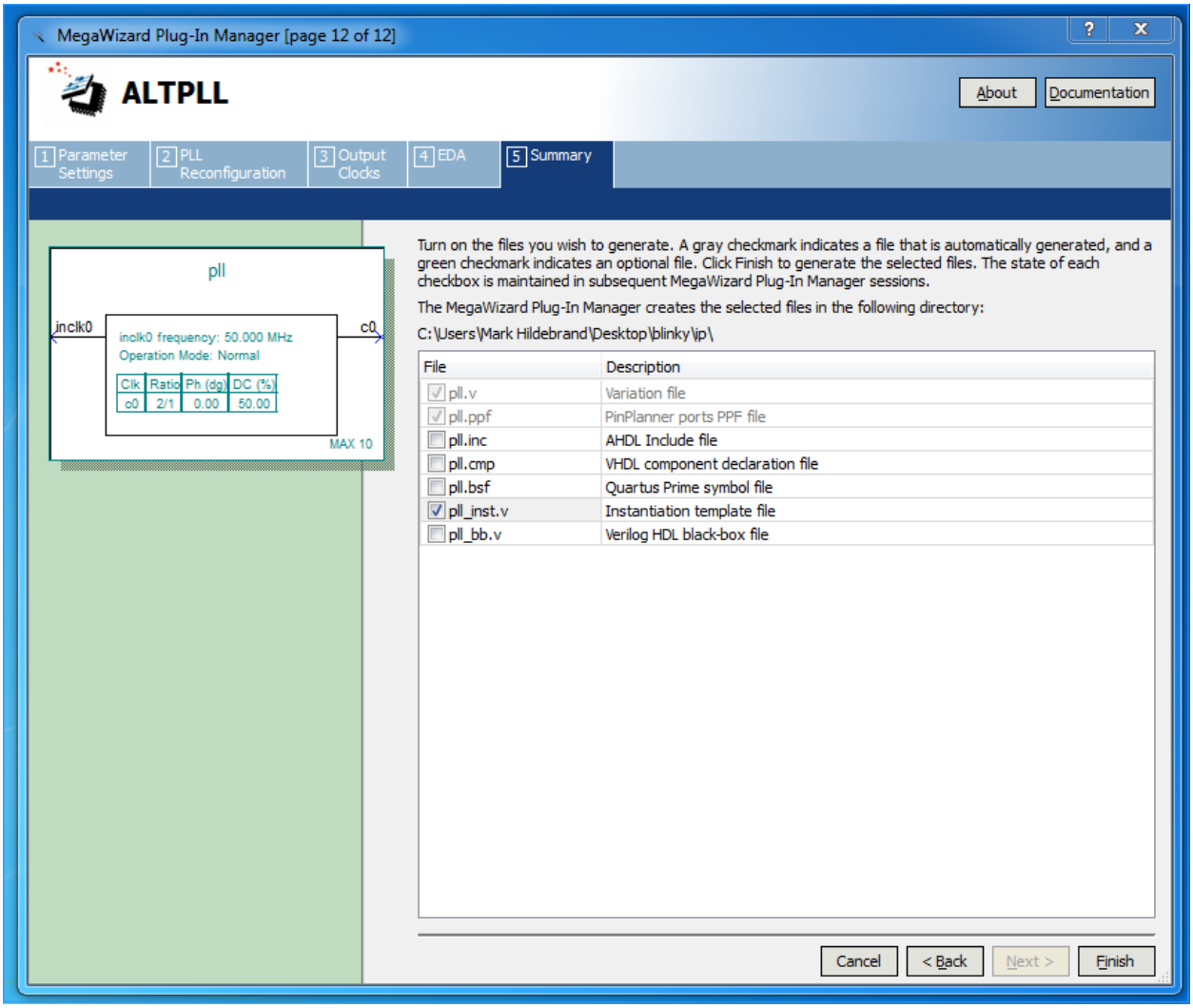
5. Click **Next**. Remove the ‘areset’ and ‘locked’ outputs. For what we do in EEC180B, you will general not need these signals.



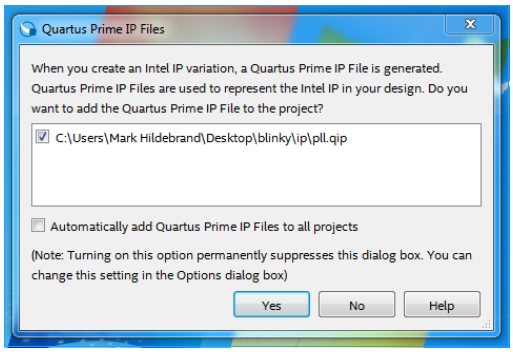
6. Click **Next** a couple of times until you arrive at the **clk c0** window. Here, you may enter the desired frequency you want in your design. Quartus will do its best to generate your desired frequency and phase. Here, the output clock **c0** is configured for 100 MHz.



7. If you want more output clocks, you may configure output clocks c1 to c4 similarly. Otherwise, keep clicking **Next** until you arrive at the final window. Make sure you check the box next to the **instantiation template file**. This will make it a little easier to use instantiate the PLL in the top-level design module. Click **Finish.**



8. A prompt should appear asking if you want Quartus to automatically add the IP component to your project. Click **yes**.

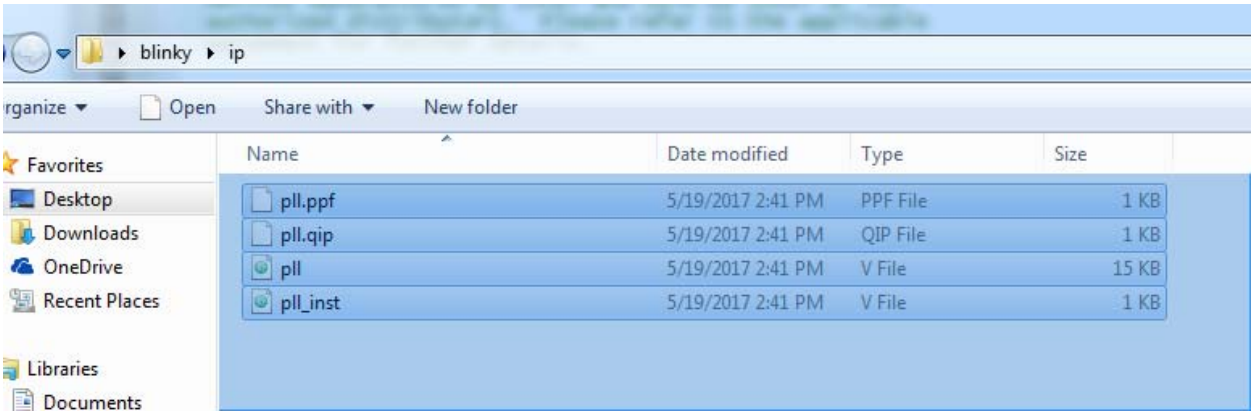


9. You should now see the file **pll.qip** in the Files section in the Project Manager window. You have successfully generated the PLL IP component.

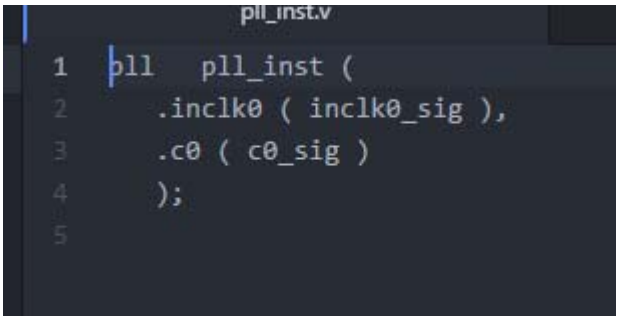
Using the Generated IP Component

Now that the PLL IP component has been generated, we need to instantiate it in our top level module.

1. Open the directory where you saved the PLL component. You should see the **pll_inst.v** instantiation template.



2. Open the **pll_inst.v** file. You will see the instantiation template for the component. Copy the contents of the file to the clipboard.



3. Paste the contents of the file in the design file where you want to instantiate the PLL (this will usually be the top level design module in your project). Rename the port maps to match the signal names in your top level module.

   In this example, the reference clock into the PLL (.inclk0) is connected to MAX10_CLK1_50. Note that the frequency of the reference clock MUST match the frequency you specified for "inclk0". On the DE10-LITE, either MAX10_CLK1_50 or MAX10_CLK2_50 could be used.

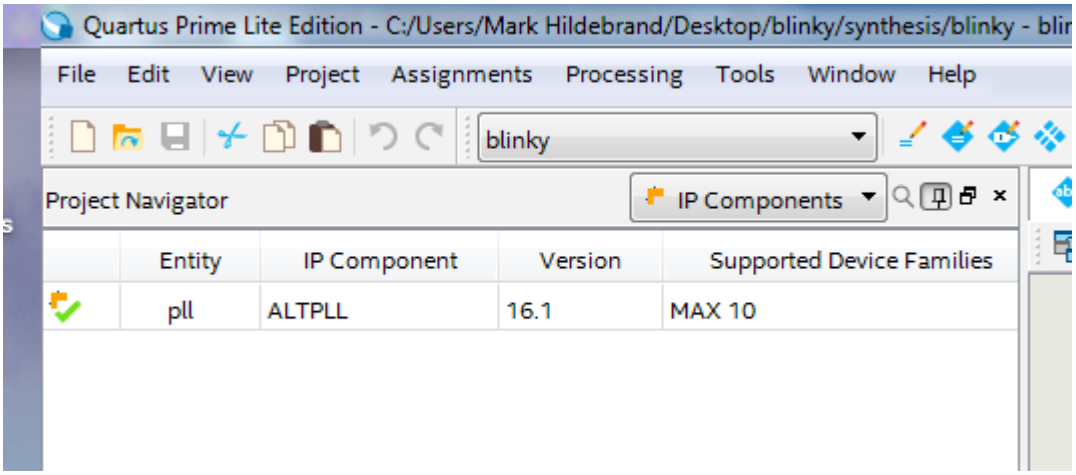
   In this example, the clock output of the PLL is named "clk" and so every flip-flop in the system should be clocked by "clk" as shown.

```
module blinky #(
    parameter NUM_BITS = 26
    )(

    ////////////// CLOCK //////////
    input                       ADC_CLK_10,
    input                       MAX10_CLK1_50,
    input                       MAX10_CLK2_50,

    ////////////// KEY //////////
    input       [1:0]           KEY,

    ////////////// LED //////////
    output reg [9:0]            LEDR
);

wire tick;
wire clk;
wire reset_n;

// Instantiate PLL
pll   pll_inst (
    .inclk0 ( MAX10_CLK1_50 ),
    .c0 ( clk )
    );


// Rename Reset
assign reset_n = KEY[0];

// Instantiate Counter
counter #(.NUM_BITS(NUM_BITS)) c0 (.clk(clk), .reset_n(reset_n), .tick(tick));

// Blink LEDs
always @(posedge clk) begin
    if (reset_n == 1'b0) begin
        LEDR <= 10'b00_0000_0000;
    end else if (tick == 1'b1) begin
        LEDR <= ~LEDR;
    end
```

4. You are done!

Modifying the PLL IP Component

If you want to change some aspect of the PLL IP component, you do not need to redo all the steps above. Instead, navigate to the **IP Components** section of the Project Manager.



Here, you will see the ALTPLL IP Component. To edit the component, simply double click on it in this window. The MegaFunction Wizard will open and you can make the changes you want. When you are done, flick the **Finish** button.

Written by Mark Hildebrand.

Versions:

2017/05/19 Written

2017/05/22 Minor changes for clarity