Lab 18: Generating clocks on FPGAs

In this lab, we'll see how to generate derived clocks properly on a AMD/Xilinx FPGAs. We use the same design as in Lab 17.

In the previous lab, the derived clock (1/5 speed of the system) was generated using a self-coded clock divider module, which was sensitive to the rising edge of the input system clock. The problem with this approach is that we could not be change the clock at half cycle, resulting in an output clock with a duty cycle of 40%.

To generate a proper half-duty clock, the best way is to use the Mixed Mode Clock Manager (MMCM) module on the FPGA.

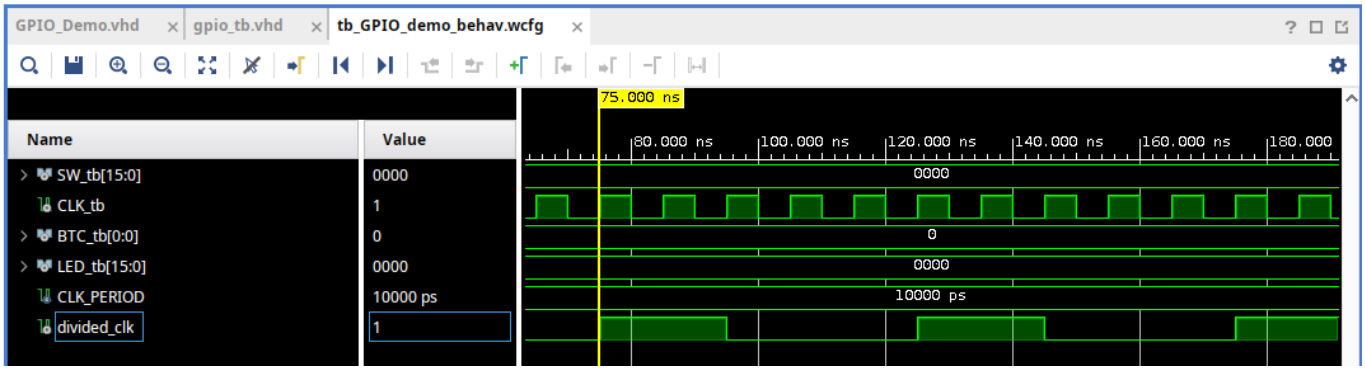
Exercise: Using the Vivado Clocking Wizard

Go to `~/labs/lab18/` and open the `lab18.xpr` Vivado project.

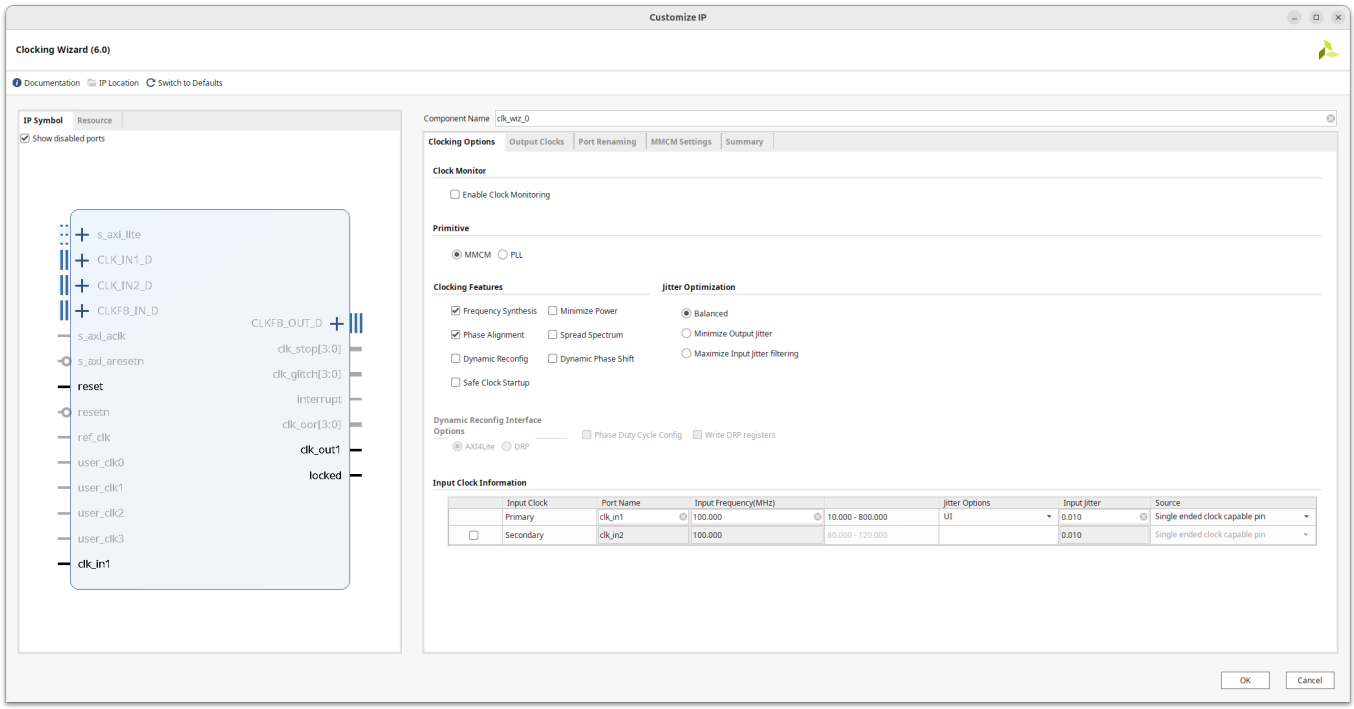
```
vivado lab18.xpr &
```

This is the same project as `lab17`. Have a look at the `clock_divider.vhd` file, to see how we created the derived clock. Run the Simulation, to see the clock behaviour. You might have to zoom in.

If you look at the `divided_clk` signal, you can immediately see the discrepancy in the time the signal is high and low.



Let's now generate our clock properly. Open the IP Catalog and search for *Clocking Wizard*. Double click on it, to open the configuration window.

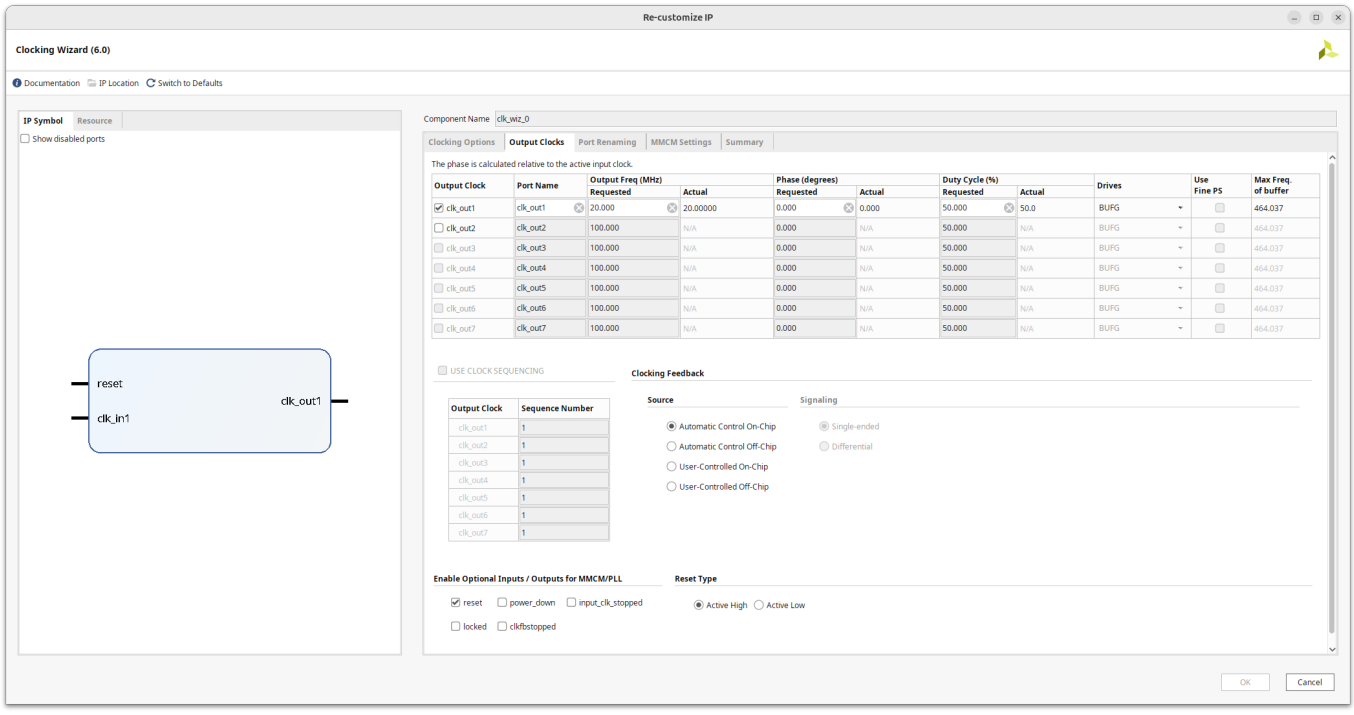


By default, it will show all the configurable port, that we don't need for our design. Untick *Show disabled ports`, to have a slimmer module.

In the *Clocking Options* page you can configure the input clock. By default, it is set to 100 MHz, which is what we are using, so we don't need to touch anything here.

Go now to the *Output Clocks* tab. There is by default one output clock enabled `clk_out1`. Modify the requested output frequency to be 20 MHz, having a duty cycle of 50 %. Untick the `locked` in the *Enable Options* area at the bottom of the window.

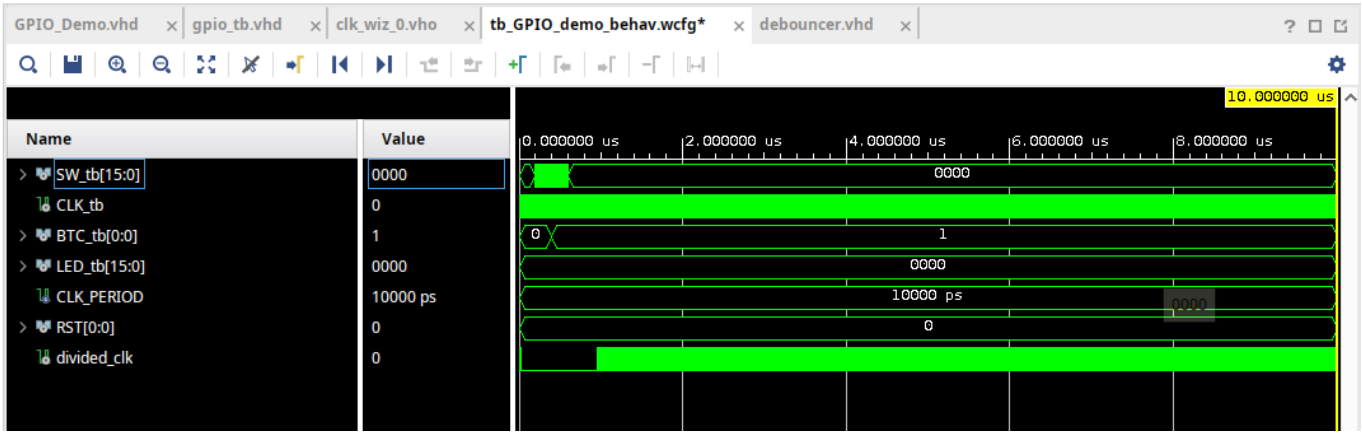
Have a look at the other options in the configuration. Once you are done, click on OK and generate the IP Output Products.



Now we need to replace the clock divider with the IP we just created. Open the `GPIO_demo.vhd` file from the Source Manager, and comment out the lines corresponding to the `clock_divider` block. Instantiate now the IP, using the generated template, as we did in Lab 16.

Save the file and run the simulation again.

You might notice that the even if the reset signal is down, it takes a while for the divided clock to be generated. This is because the simulation is accurately modelling the MMCM, and it takes a few microseconds to stabile, before starting to output the clock. This would have been signaled by the `lock` signal we disabled. Zoom in the wave, to actually verify that the generated clock has a period of 50 ns and is half-duty.



Generate the bitstream and load it to the board, and check its functionalities.