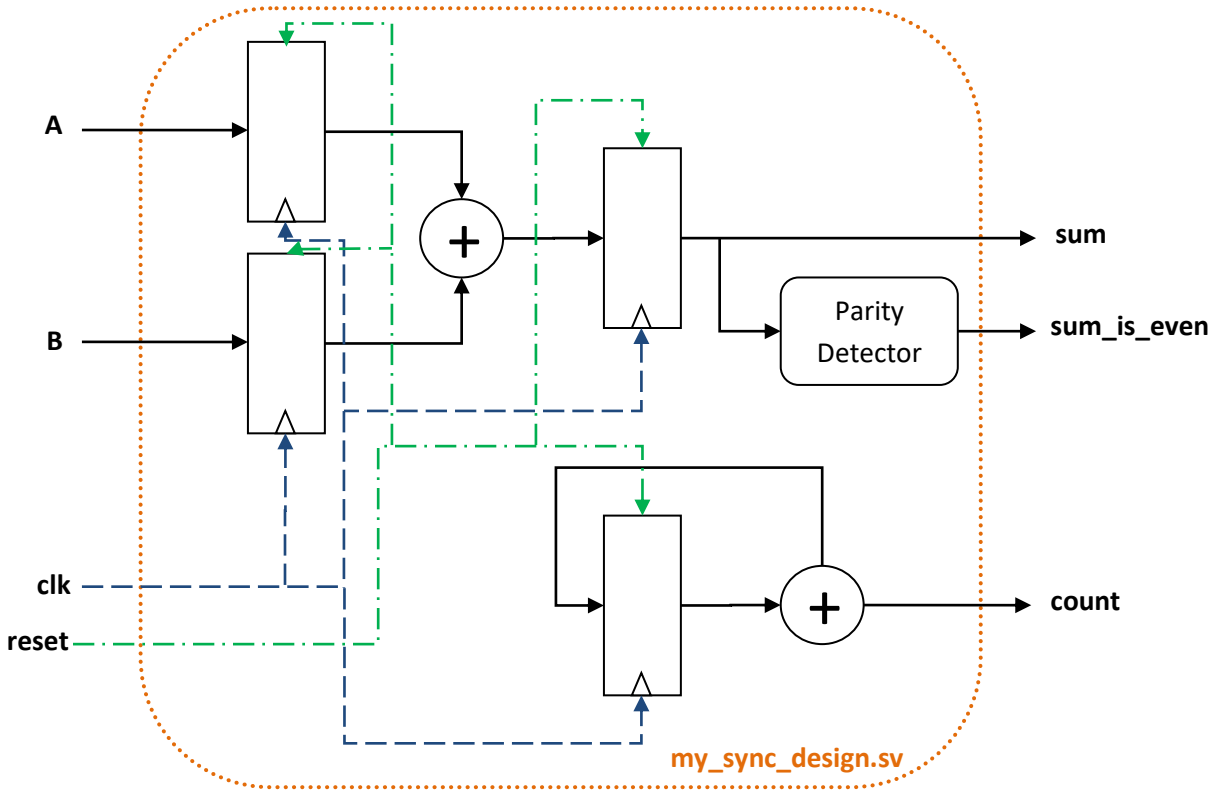
Introduction to synchronous designs PART B

As already showed in part A, here is the case study for today: a simple adder, a parity detector at the output, combined with a clock counter (commonly called a timer, but hey!):

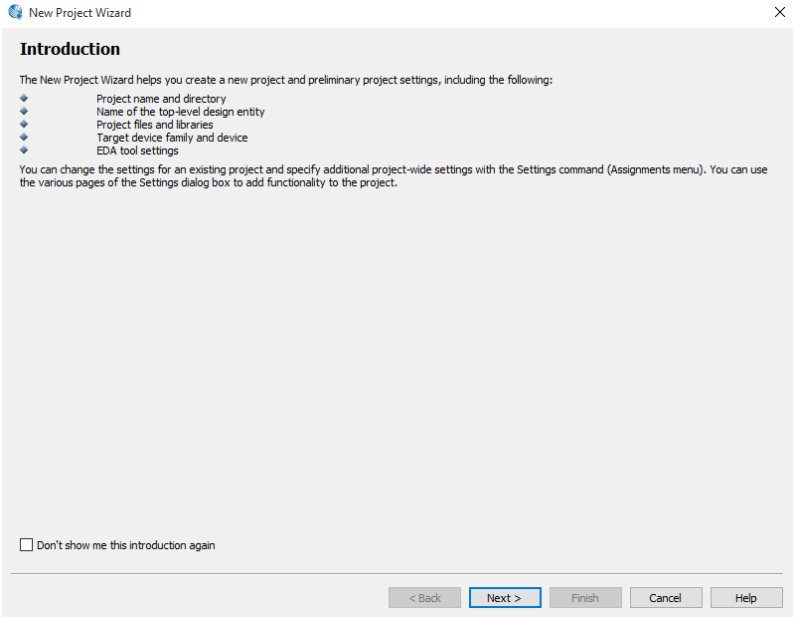


The Verilog design of this circuit can be found in "my_sync_design.sv" (see Moodle, "Lab 2" directory). Please take a moment to look through this file, and understand its design. The other file given is the testbench file; please take a look at it now also.

Now, let's start using Quartus!

1. Locate Quartus on your system, open it up. You may be greeted by a "welcome pop-up" asking you what you wish to do next: close it. Now let's create a new project using "File (menu) > New Project Wizard". Please note that the snapshots in this document were taken from the Quartus 15.0 version, but as there are only small variations compared to Quartus 18.0, we did not retake them.

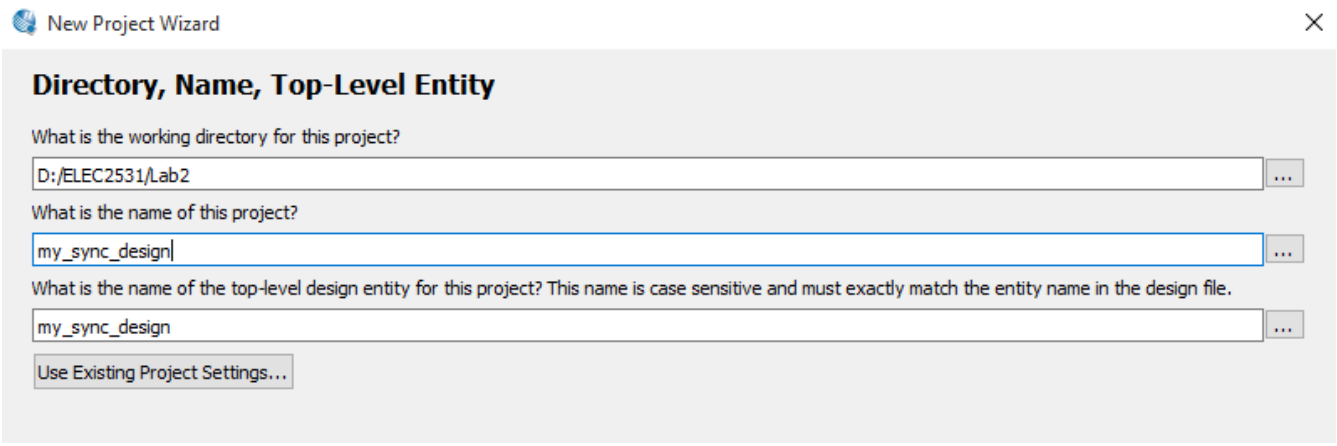
   Here is what it should look like once opened:



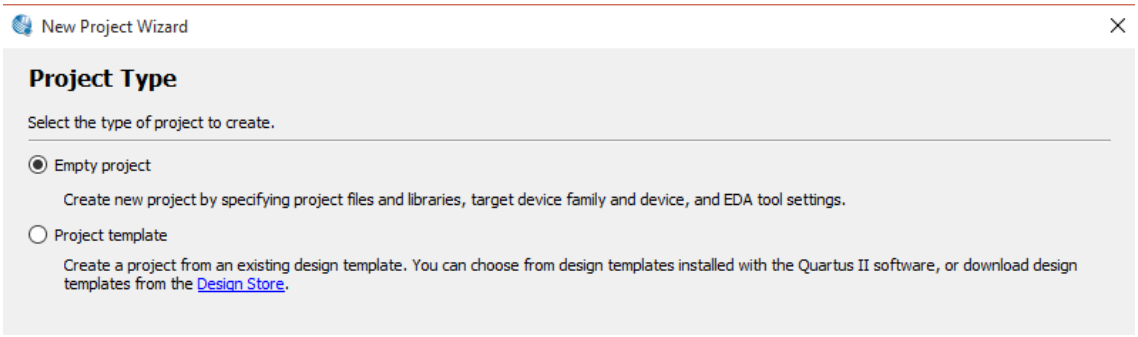
2. Click “Next” on this introduction page.

3. Now, before going further, copy the two SystemVerilog files we provided (“my_sync_design.sv” & “testbench.sv”) in a new directory, which, as you are now well aware, **must contain no spaces or special characters in its path**. Also let’s point out that it **must not be located on a “cloud” folder** (Dropbox, OneDrive, etc.) **or on a remote one** (e.g. your Oasis repository).

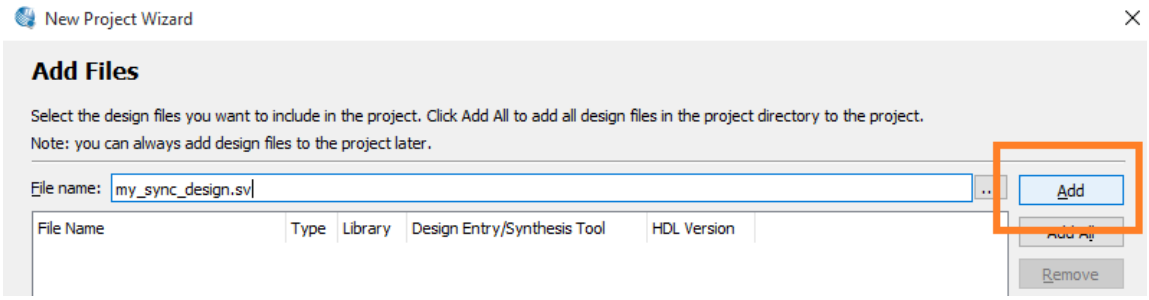
   Then, get back to the Quartus project “wizard”, set your working directory as in the figure below and specify a nice project name in the second field as well as your top-level module (here “my_sync_design”) into the third one. Please note that these last two can be the same (as we did in our example), but the third field must absolutely match the name of your top-level module:



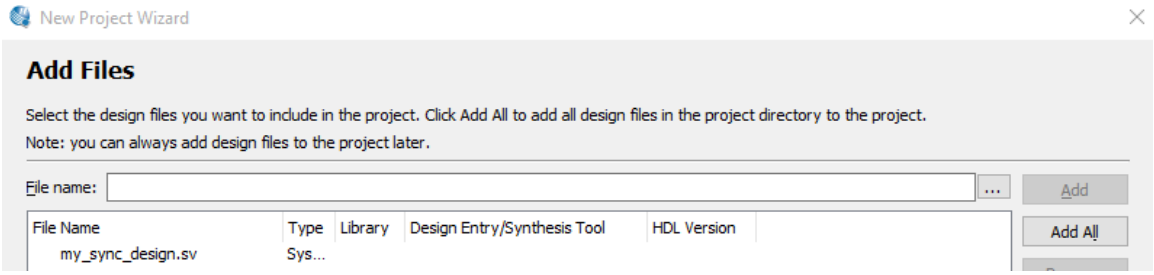
4. Click Next. On the third step, let the “Empty project” box selected and click next again.



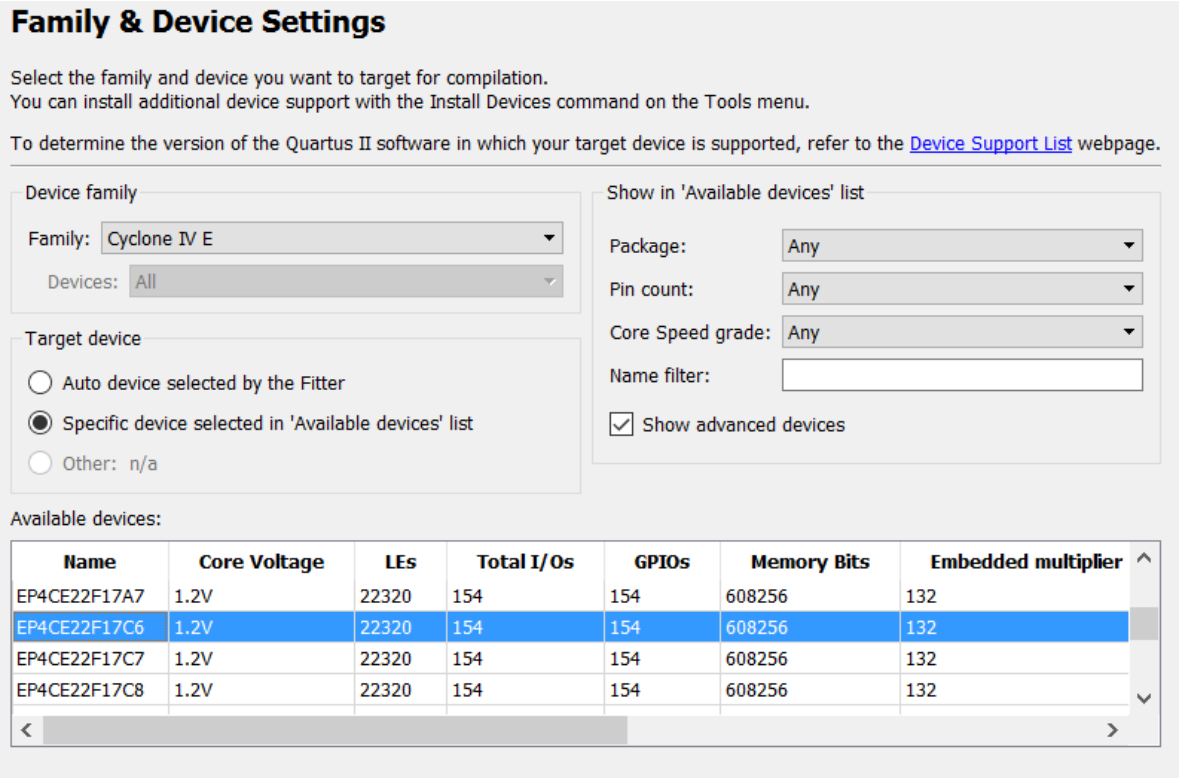
5. Next, add the Verilog files to your project. You should not add the testbench file here, that’s used only during the simulations. Let’s recall that a testbench is used to simulate external stimuli in order to verify circuits during their design and is not part of the actual circuit.



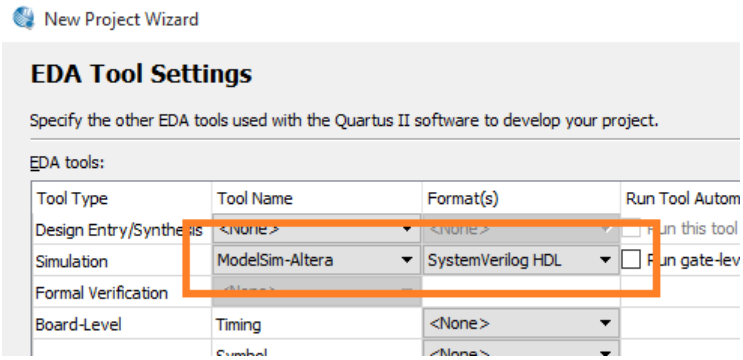
Before clicking next, be sure to have well added your Verilog file by clicking the button “Add” after having selected it. Here is what it should look like before you click next:



6. The next step allows you to select the FPGA you want to synthesise for. This year, as you may already now, we will use a Cyclone IV E FPGA, model EP4CE22F17C6N. However, we won’t use them before next week so, the device does not really matter here as we won’t program an actual device. But still, choosing the right device here will give you a relevant compilation report in step 19.



7. Next, you will be asked if you want to use third-party tools for specific tasks in the synthesis flow. We only want to set the simulator, in order to use ModelSim (the one you already know and love) with the “SystemVerilog” format:

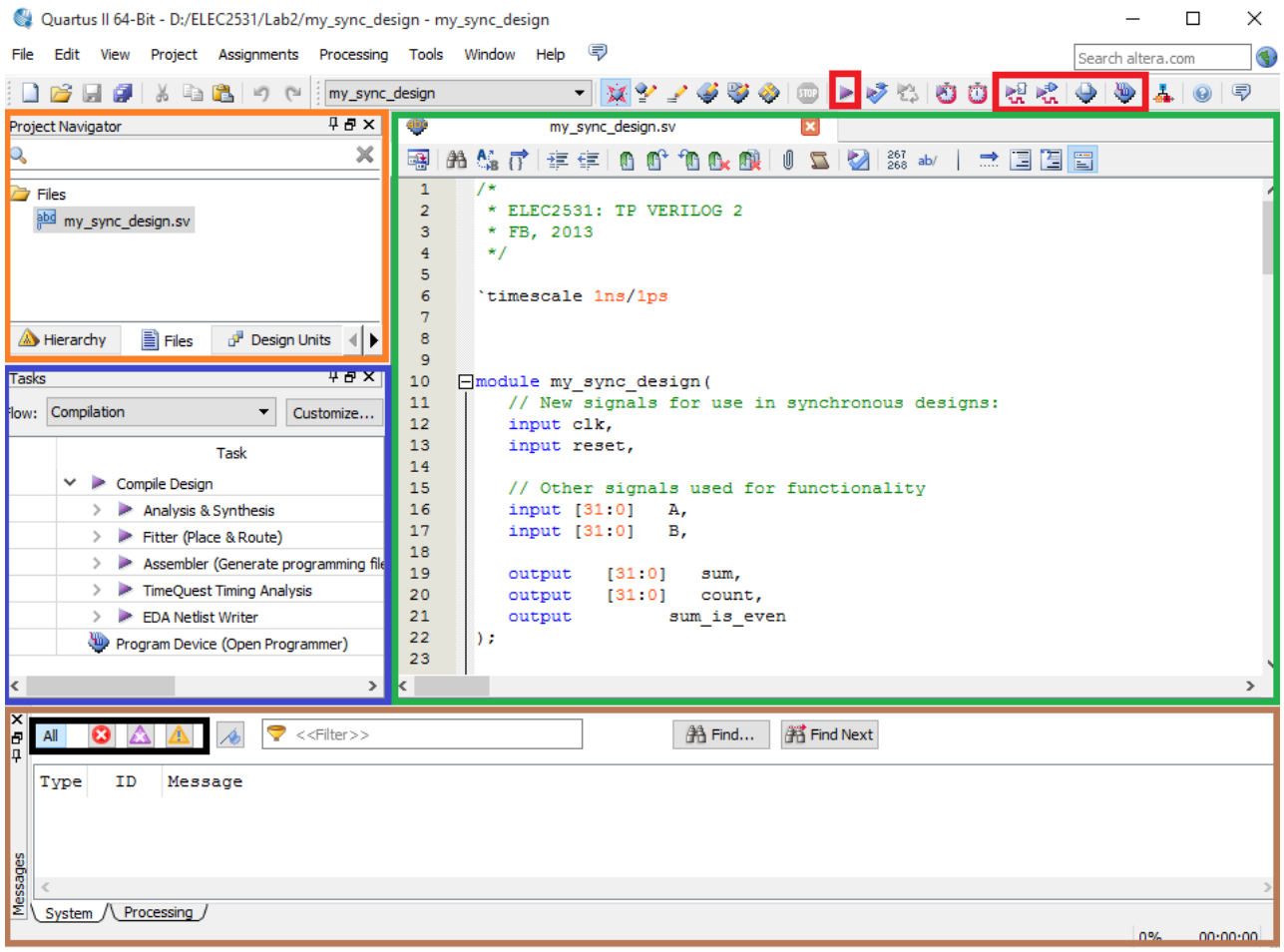


8. Next -> Finish, project created!

9. Let’s take time to discover properly the graphical user interface. It is organized in quite the same fashion as ModelSim: you have several sub-windows that you can make appear or disappear using the menu (“View > Utility Windows > …”). Unlike ModelSim however, these sub-windows cannot really be reorganized and undocking them from the main GUI is not really convenient.

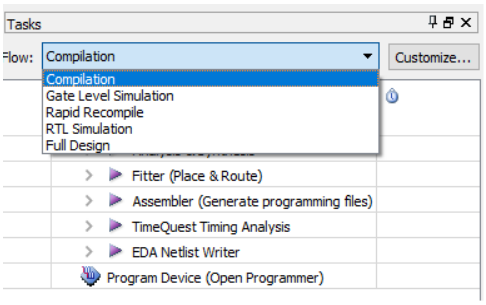
First, we have the "**Project Navigator**" from which you can consult the hierarchy of your project and its composition either in terms of files, design units or IP components. On the top bar, we have **various buttons** which are useful shortcuts: the most useful are the ones highlighted (compilation, RTL sim, Gate Level sim, Compilation Report and Programmer). Next, on the left, the "**Tasks**" sub-window gives you information about the different synthesis steps (during and after compilation of your design). It also allows you to launch parts of the compilation separately (using a right click on each step + "Start"). On the right, you simply have a **text editor** where, for example, you can modify your Verilog files or consult compilation reports.

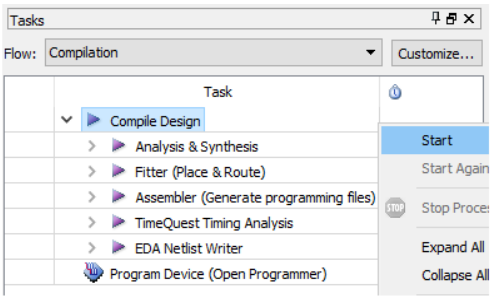
The last default sub-window is the "**Messages**" one, at the bottom. It gives you feedback during the compilation. There, you can discriminate what is displayed using the **buttons** highlighted. First, you have the "errors" that will prevent the compilation from completion and hence require you to deal with them. Next, the "critical warnings" won't stop the compilation, but you should carefully consider them, especially if you plan to program a physical device (in the sense that your design might simply not work, not that it could damage the device). Finally, you have the "warnings" which should always be read (especially at the first compilation) but that you can> generally skip. Finally, Please note that you can find other sub-windows in the menu ("View > Utility Wind. > …"). For example, a Tcl command line if you are prone to script your design flow.

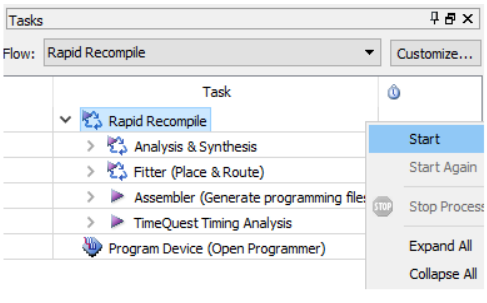


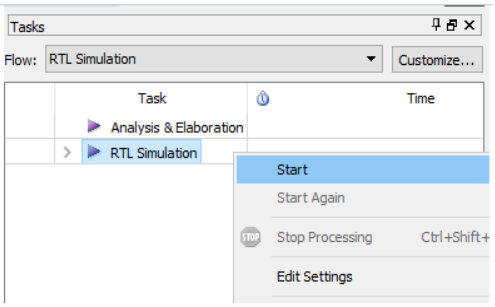
10. Now, let's inspect a bit more in details the "Tasks" sub-window. If you scroll its "Flow" menu, you will see that there are several possibilities, composed of more or less steps. The important thing to understand here is that you do not need and should not do a full compilation every time you want to simulate your circuit or program your FPGA. Understanding this now and applying in this in the future will save you a great amount of time, trust us.

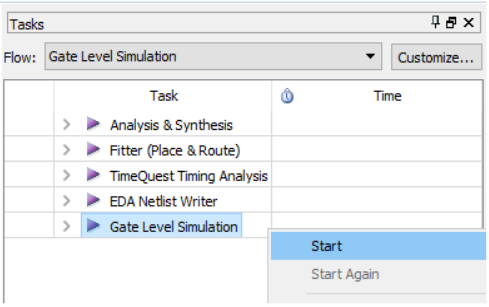
    a. **Compilation**: will go from the "Analysis" to the "EDA netlist writer" (that produces the physical netlist). It is the full compilation (and hence the more time-consuming flow), when you want to go all the way from your behavioural netlist to the device programming.
    b. **Rapid Recompile**: very useful to recompile after small code modifications, especially when you have large designs (or slow computers). However, have in mind that it is only available in the licensed "Standard" version of Quartus (installed on the lab room's computers) and for more advanced FPGAs (that we will normally use with ELEC and ELME students next semester with another board).
    c. **RTL Simulation**: here, the flow is limited to what's needed for a behavioural simulation (using ModelSim), namely the "Analysis & Elaboration".
    d. **Gate Level Simulation**: same principle as for the RTL simulation, but adding the circuit "Synthesis" and "Place & Route" steps. Please note that the "TimeQuest Timing Analysis" is not mandatory and will not actually be covered in this course. However, for the ELEC or ELME students, this might be useful for your respective ELEC2103 project to look for timing failures in larger designs.
    e. **Full Design**: should not be useful to you, given that it contains many stuff out of this course's scope.



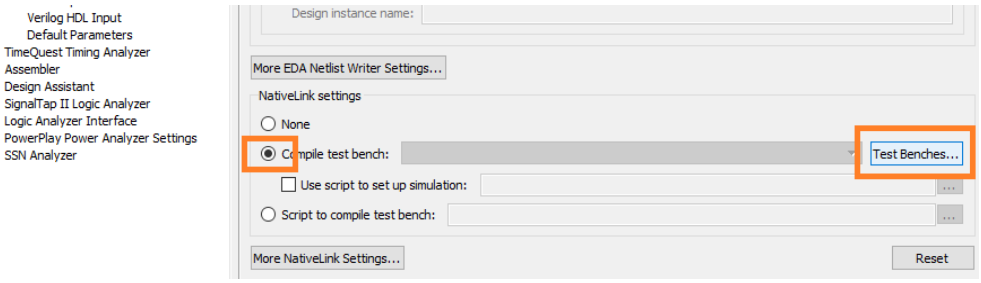




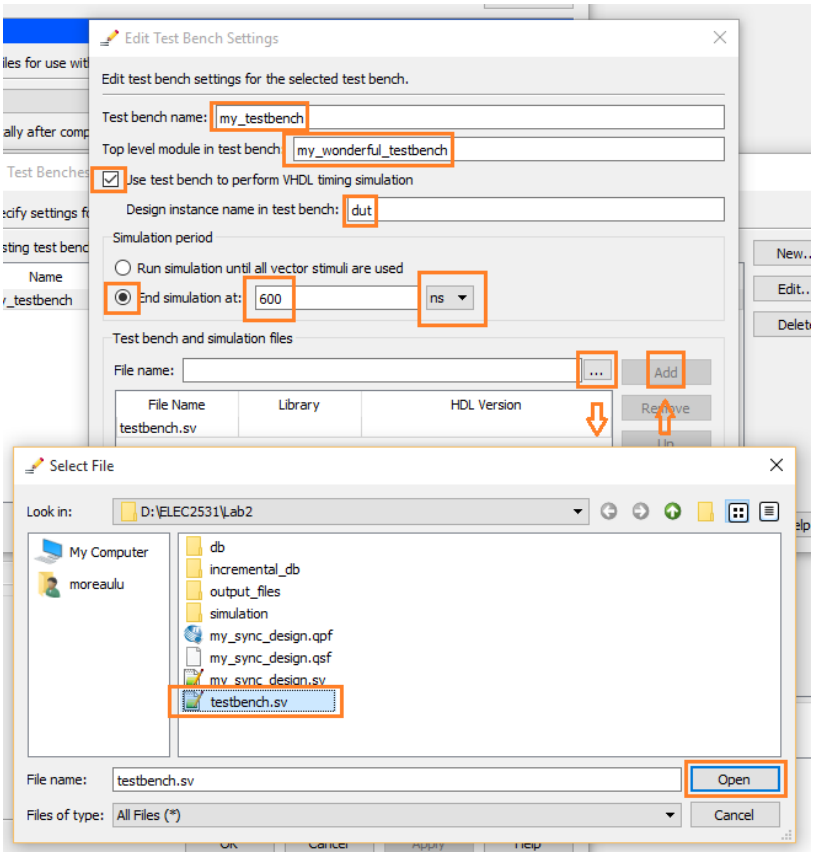




11. Now, the first thing to do with a new circuit implementation is to verify its behaviour using RTL simulation. Obviously, we will use ModelSim to do that. Let's see how to configure Quartus in order to launch ModelSim from it: on the menu, go to "Assignments > Settings > EDA tool settings > Simulation". Check the "Compile test bench" box and create a new Testbench using the ad hoc button:

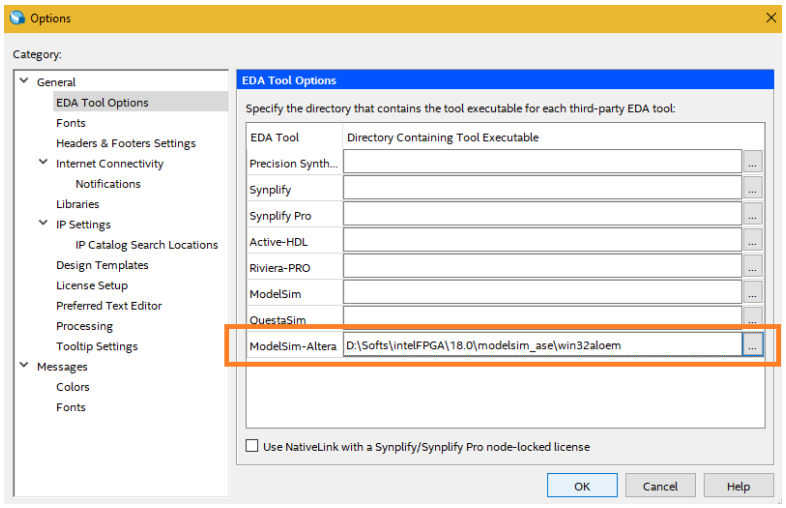


Another window should pop-pup. Click on "New…" and then fill the other pop-up as below:

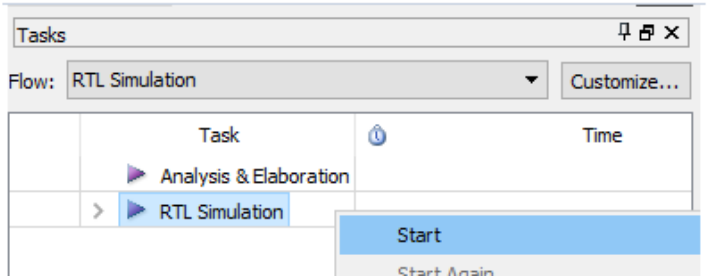


Here are a few comments: the testbench name is not important, while the "top-level" module name have to match the one in your testbench file. Besides, if you don't set a simulation period, ModelSim will simulate indefinitely and you will have to end it manually which is really not convenient. And finally, you obviously have to specify the testbench file, using the browser. **Don't forget to press the "Add" button !!** If not, it won't be considered. Click 'ok' until you get rid of the clutter of windows.

12. Before getting to the actual simulation, we have to check that ModelSim is well linked in Quartus. In the menu, click on the menu “Tools > Options”. The pop-up window showed below will appear. In “General > EDA Tools Options”, set the path to the “win32aloem” folder in the ModelSim installation directory (“intelFPGA\18.0\model_sim_ase”) that obviously differs in location from one computer to another (for the didactic computers, it is simply on the C:\ root).

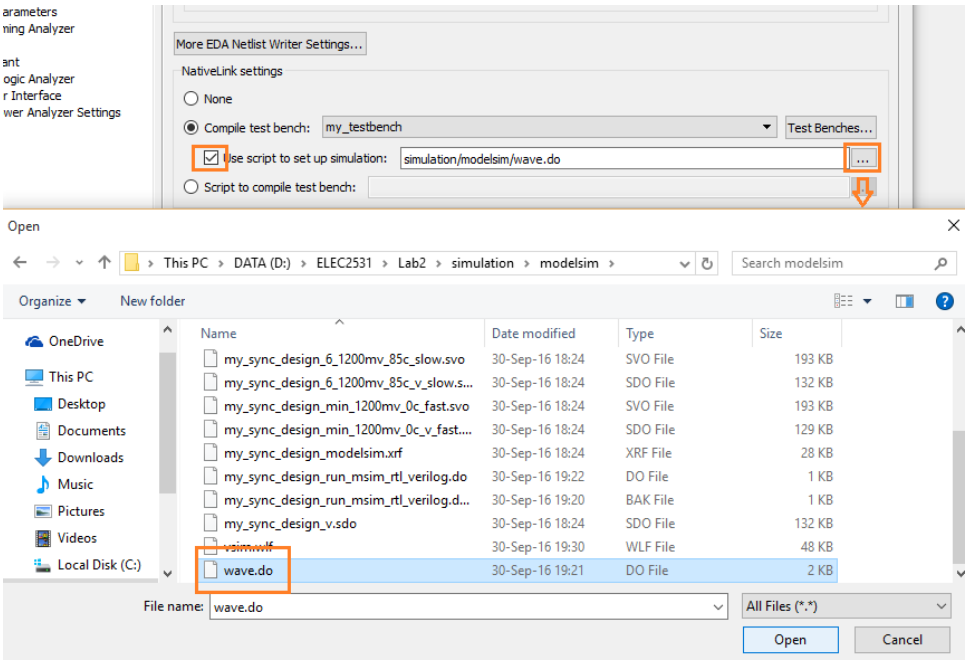


13. Now, on the “Tasks” sub-window, set “Flow” to “RTL Simulation”. Then, have right click on “RTL simulation” below and click on “Start” (see the figure below). This will run the “Analysis & Elaboration” step and then automatically open ModelSim to run the simulation. Let’s point out here that once the “Analysis & Elaboration” is already done, you can also launch a behavioural simulation via the menu “Tools > Run Simulation Tool > RTL Simulation”.



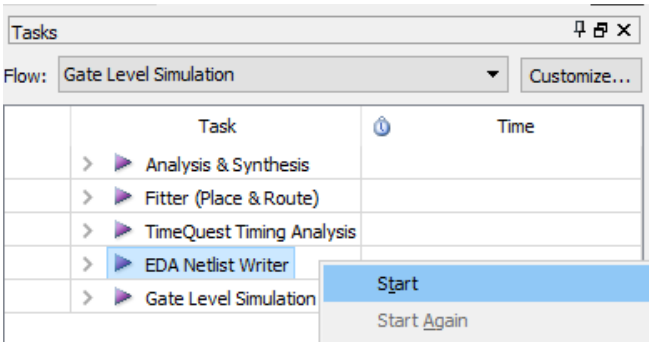
14. Look at the simulation waves and, based on the block schemes, check that the circuit is functional and achieves what it is supposed to.

15. Another important feature of this testbench configuration on Quartus: you can also directly set it to use a Tcl simulation script like the ones you learnt to build during the previous lab session. Let’s make a quick test: on the ModelSim “Wave” sub-window, set the proper radix to the different signals and different wave colours, then save the configuration on a “.do” file using the menu (“File > Save Format”). Then, close ModelSim and get back to Quartus, on “Assignments > Settings > EDA tool settings > Simulation” and add your script like this:

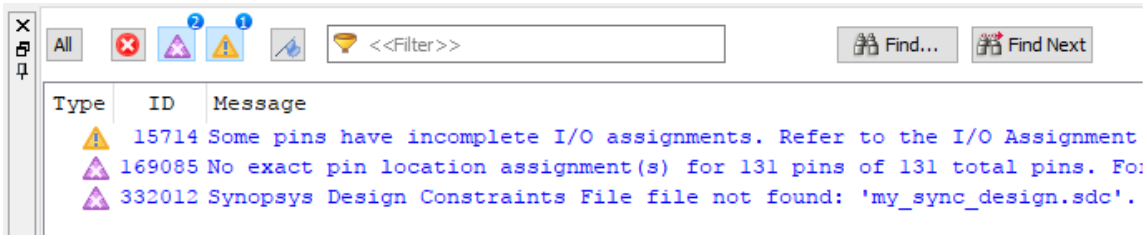


Press the “STOP” button on the toolbar to end properly the previous simulation. Then re-run the RTL simulation to check that your script was well taken into account. Normally, ModelSim should not run any simulation time if you didn’t add any “run” command in your “.do” file. Therefore, run 600ns manually (using the “Transcript” or the toolbar).

16. Let’s now perform a post-layout or gate level simulation. To do so, we need the circuit to be synthesized and fitted by Quartus. Hence, on the “Tasks” sub-window, set the “Flow” to “Gate Level Simulation”. Before trying to launch the simulation, have a right click on “EDA Netlist Writer” and click on “Start”.

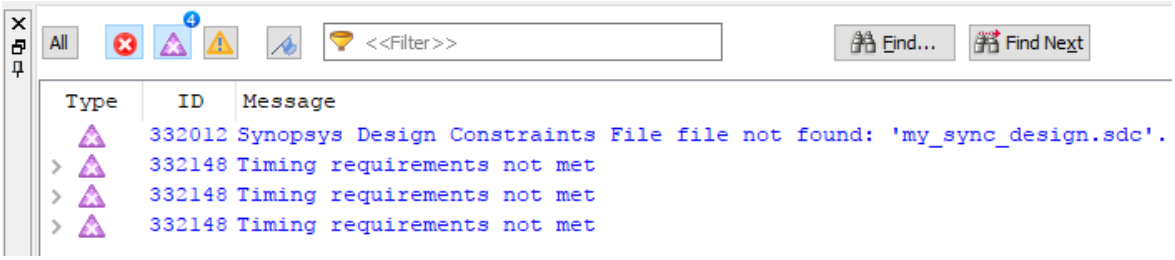


17. The different steps should be processed without errors. However, you should have two critical warnings and a warning, that you can all ignore:

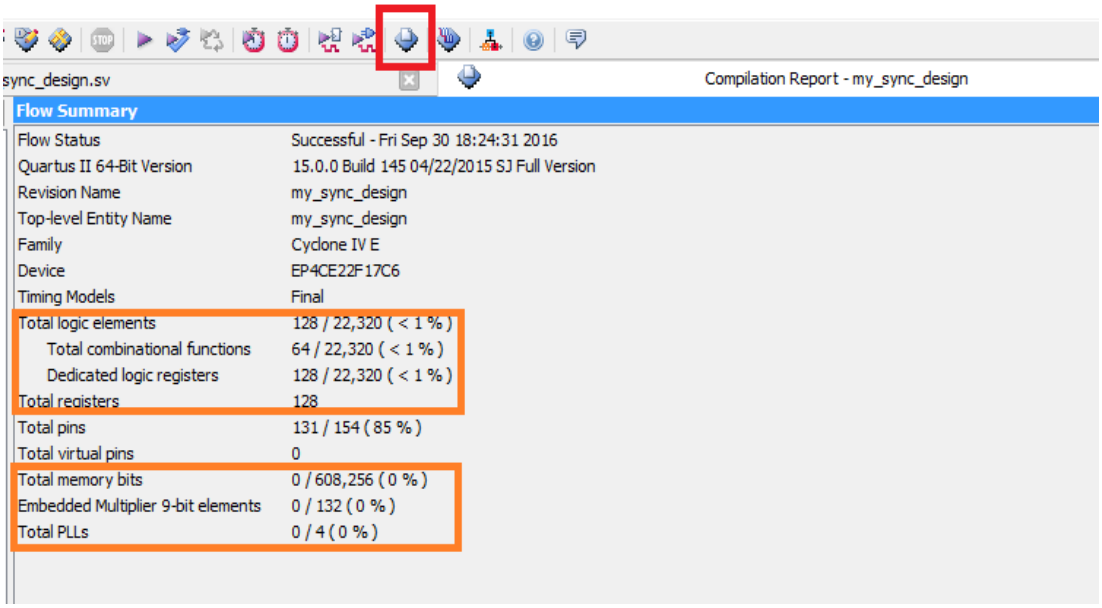


The first two are related to the fact the we have not set the pin assignment (more on this next week) but it does not matter since we will not program a device today. The last one is related to the fact that we did not specify the timing constraints of our design and therefore it cannot check the timing requirements (out of the course's scope, see ELEC2570 for more on this).

18. As you probably noticed, the "Timing Analysis" step was not performed by the tool. You can do it manually now (right click + "Start"); it should give you the four critical warnings below that you can ignore. Note this analysis is always run when you launch a full compilation on Quartus. However, let's recall that even if it can be useful to detect timing errors we won't deepen this tool usage during this course.



19. Take a moment to check the compilation report, which should have opened automatically in the text editor (if not, you can make it appear using the ad hoc button, using the menu "Processing > Compilation Report" or using the "CTRL+R" keyboard shortcut). Especially, it informs you about the usage of logic elements, registers, memory bits, multipliers and PLLs. These are different circuit elements (and finite resources) that your design can use to run on your FPGA (you will get familiar with them later on the course). One of the synthesis purposes is to translate your behavioural (Verilog) netlist into a netlist composed of instantiation of these elements. This report will be particularly useful for ELEC and ELME students in their respective project.



20. Now, let's do the post-layout simulation. This kind of simulation is closer to the reality than the previous one because the component timing and delays are taken into account. However RTL simulation is way more practical to verify the pure design behaviour and functionality, so it should always be the first step.

    You can launch the post-layout simulation, either on the "Tasks" sub-window or using the menu "Tools > Run Simulation Tool > Gate Level Simulation". Please note, that using the "Tasks" sub-window, the tool may not propose you to choose a model (check both possibility to see the difference).

    Once it is open (choose the for example "slow 85" model[1] first and then the others), observe the waveform, and try to see the differences. You should observe:

    a. Timing differences. In->Out is slower.
    b. Glitches: why and where?
    c. Functionality still OK?

That's all folks, enjoy!

[1] « slow » and « 85 » refer to process and temperature corners. These are out of this course's scope (see ELEC 2570 next year).