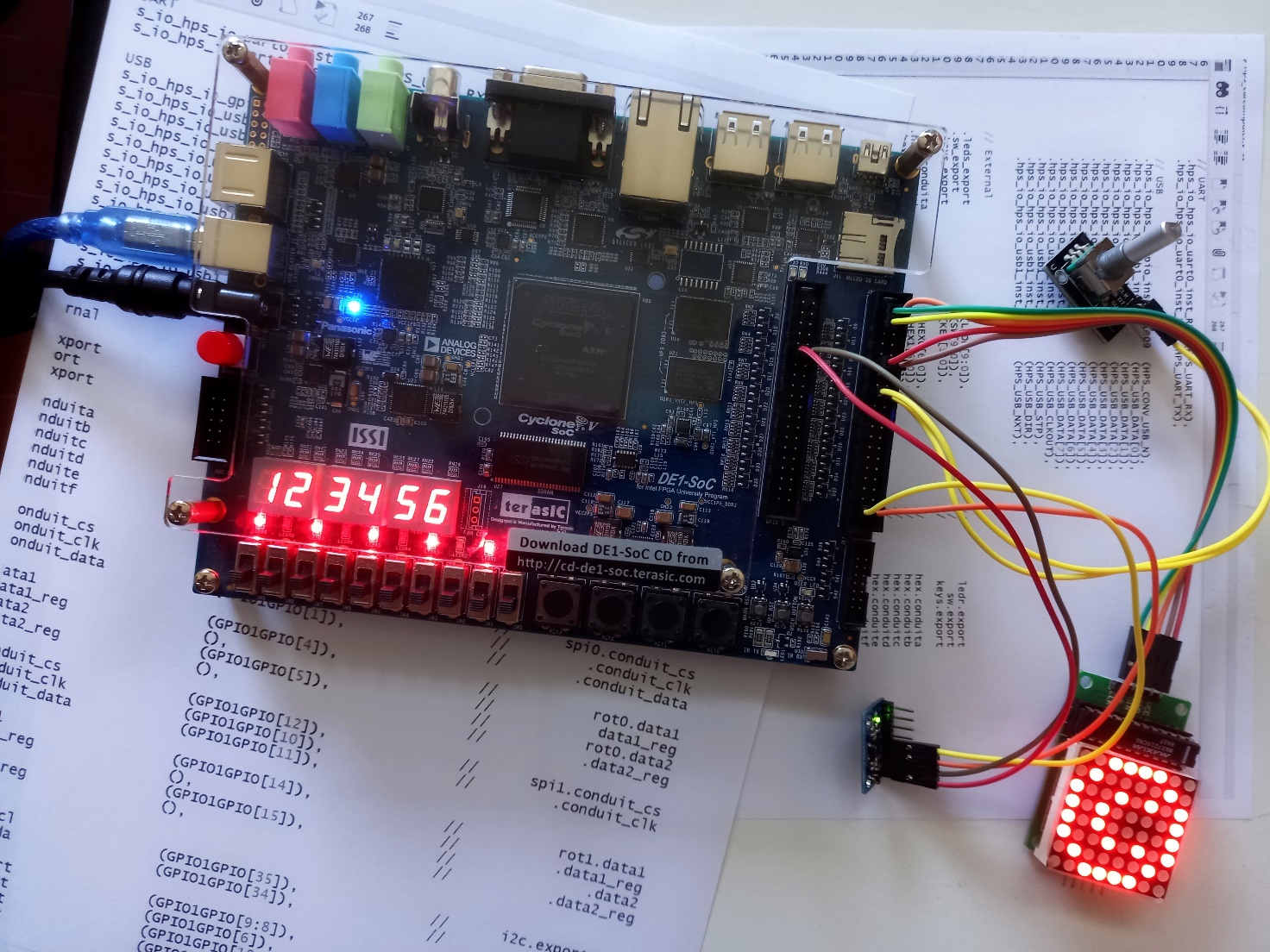
Electronic Systems Design

# Prof. Marsi Stefano - University of Trieste Academic Year 2025/26

Tutorial 1



**Introduction to developing a simple complete project.**  
**Hardware** used: Terasic DE1-SoC Board  
**Software** used: Quartus 22.1

Tutorial 1

Realization of a simple logic circuit on DE1-SoC

Description: A simple logic circuit is created on DE1-SoC where inputs are driven by the switches and output drive the LEDs.

Purpose: to familiarize with the development tool, with the development board, learning the basic steps to follow to complete a complete project.

Expected learning:

* First rudiments to navigate within the quartus tool
* Compilation of a project described in schematic form
* Building a described project using Verilog HDL
* Imposition of some implementation constraints (Pin position)
* Circuit simulation
* Circuit layout display
* Download and run the circuit on the DE1 board

# Procedure

1. Introductory phase

Open quartus tool via the icon on the desktop or from the list of installed programs.

Create a new project either by clicking on the appropriate button or by using the

*File > New Project Wizard*

*(Next)*

Define

* the directory in which to save the system files (the use of spaces or unusual characters is not recommended – so avoid for example directors such as "My Documents") moreover some system directories could prevent writing from external programs.   
  The suggestion is to create an “on purpose” directory like : **C:\QuartusProjects**
* the name of the project, and define (provisionally)
* the name of the top level block in the ranking of the project (Top Level Define Entity) – typically this takes the name of the project itself but it can be changed later.

(Next)

If you want to import files or settings from other projects, you can use this page clicking on “Use Exsisting Project Settings” button. In this first example, however you can skip to the next page

(Next)

Define whether the project is developed from scratch or from a pre-loaded template. In this first tutorial, we start a blank project

* Empty Project

(Next)

Eventually you can include pre-existing files that in this first case have NOT been developed yet

(Next)

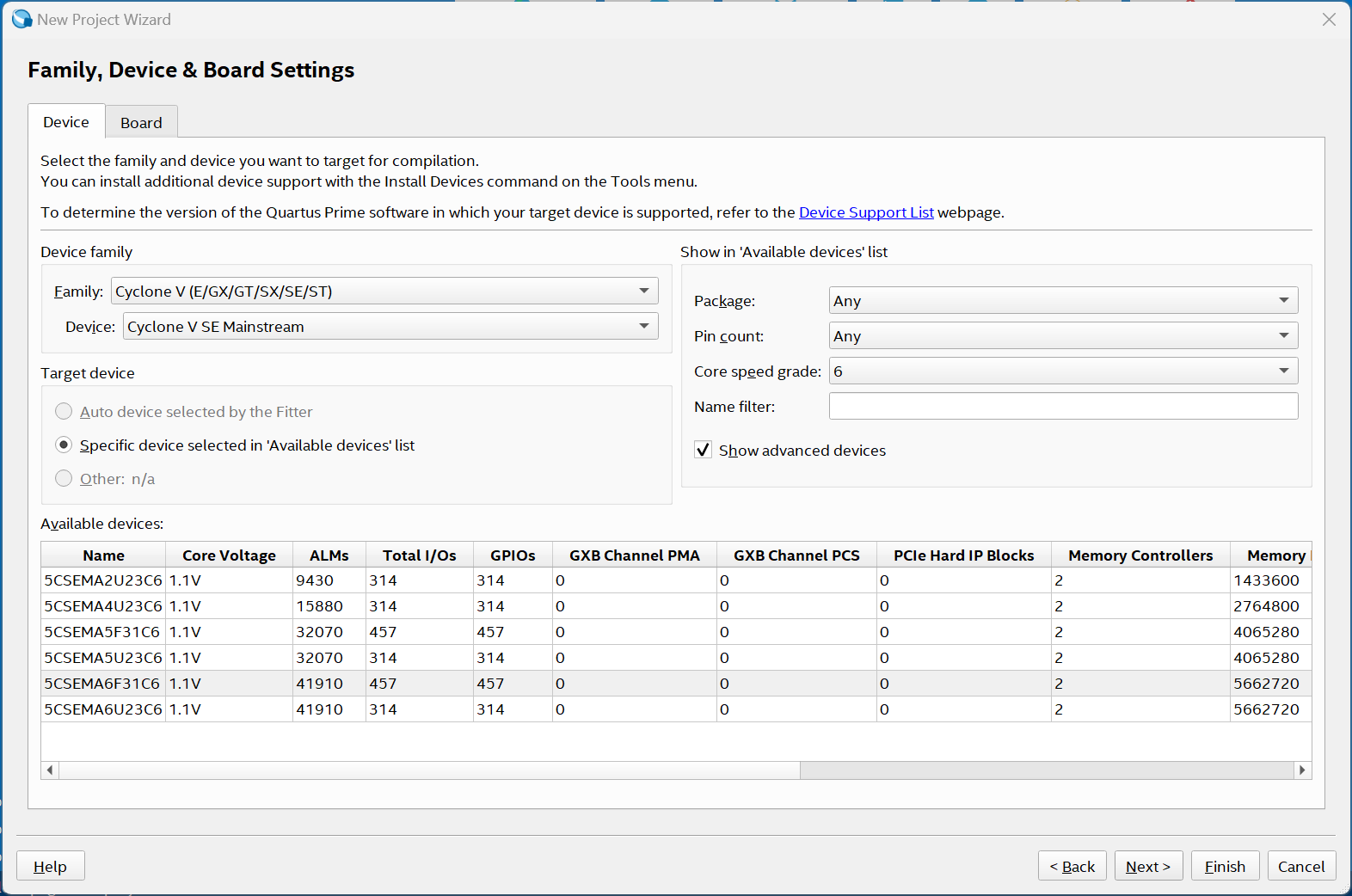
Define the device (FPGA) on which to develop the entire project. Of course, the design itself, the available resources, and the structure itself depend on the device used – there are hundreds of FPGAs that are different in family, characteristics, performance, pinout, etc. **It is ESSENTIAL** to choose the correct FPGA. In our case, a 5CSEMA5F31C6 of the Cyclone V family is mounted on the DE1-SoC, so we will choose the:

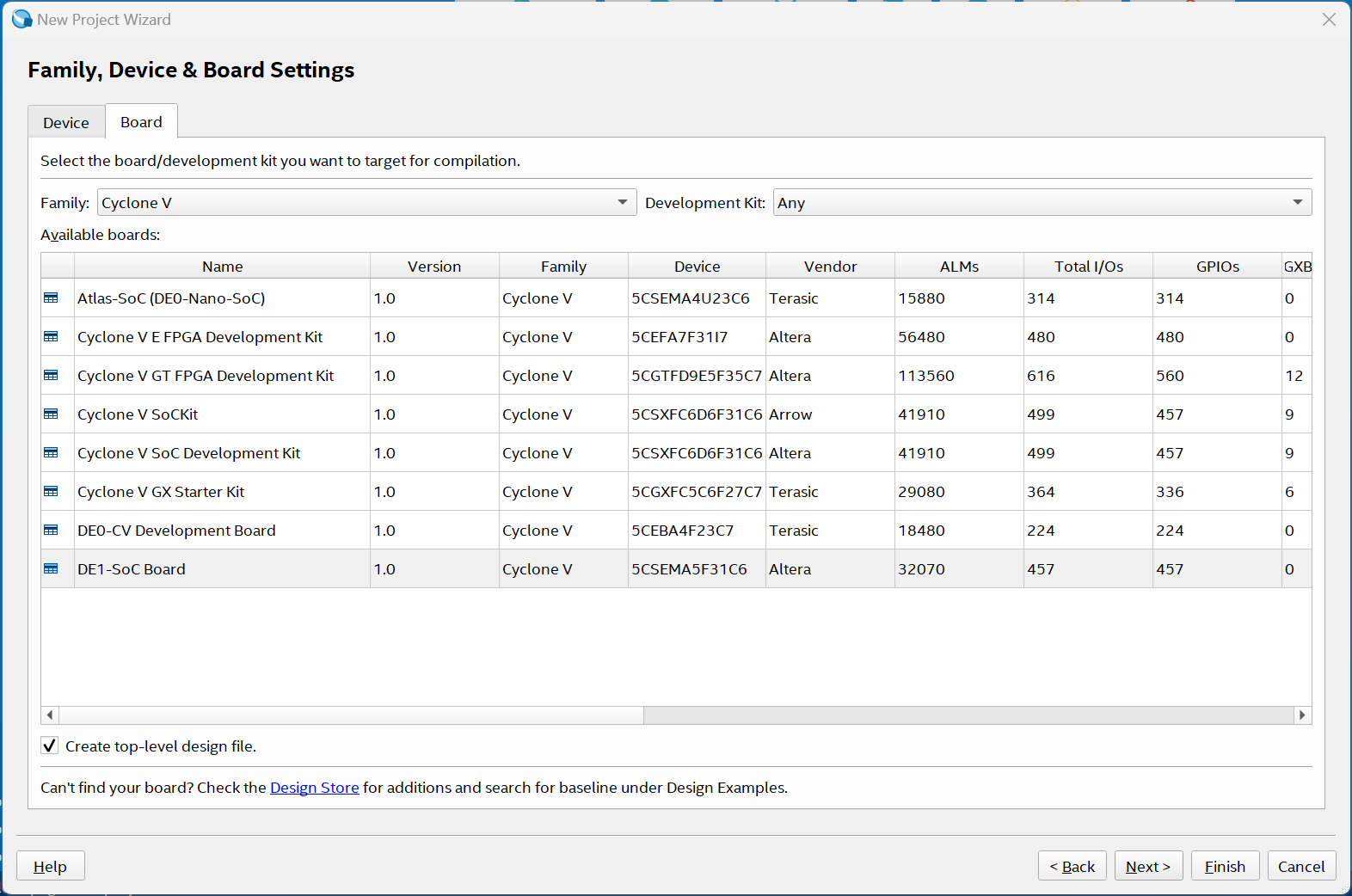
Family: Cyclone V (E/GX/GT/SX/SE/ST)

Device: Cyclone V SE Mainstream

* 5CSEMA5F31C6

You can help with the search by using appropriate filters, or by using the Board subfolder (Available if you have installed the Intel Academy University Program – which offers, for some development boards, by default the device mounted on them.

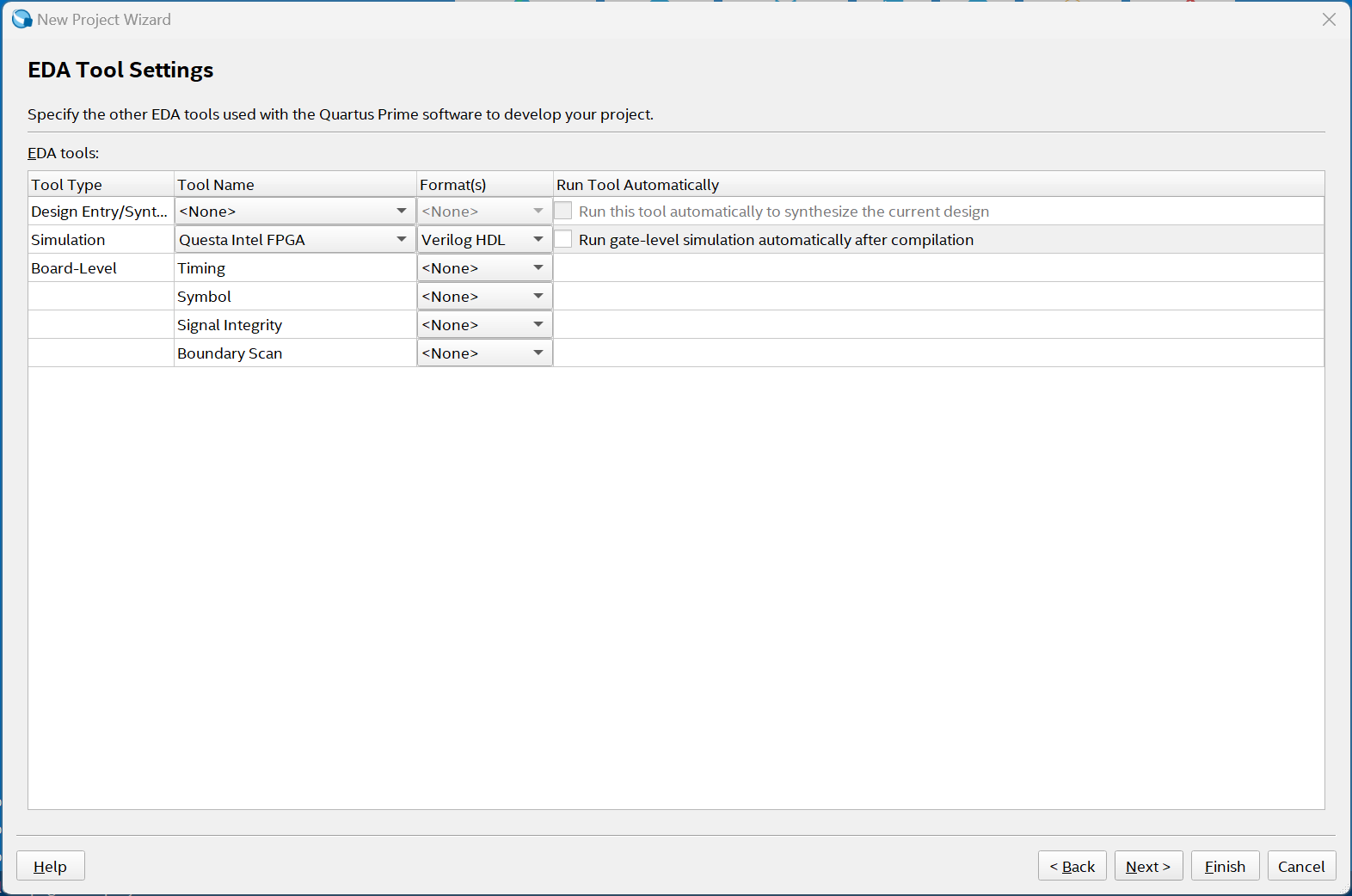




The table on this page allows you to compare the performance of the various devices (logic cells, available memory, integrated multipliers, PLLs, I/O resources, dedicated clock lines, HPS Cores) ... this could be useful when the device on which to build the system has not yet been defined.

(Next)

On this page you can set up third-party programs to be used for synthesis, simulation and temporal analysis. In this case, define “Questa Intel FPGA” as the simulation tool and Verilog HDL as the language to be adopted – disable "Run Gate-level simulation automatically after compilation".



(Next)

A summary page lists the choices made, in order to verify their consistency

(Finish)

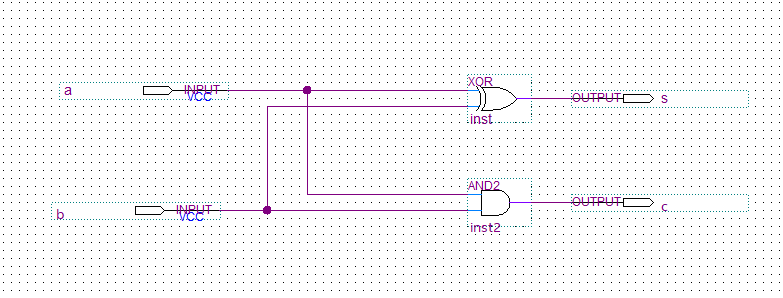
#### Definition of the system architecture (Schematic) [5-6]

File > New

Block Diagram / Schematic file (OK)

Suppose we want to develop a "Half Adder" capable of generating sum and carryover using two input bits

On the new graphic page that appears, draw the schematic of the system to be implemented similar to what is shown in the figure.

Use the  and  keys to access the logic gate library and to create the various shortcuts. 

In particular, it is suggested:

- to view the different logic gates and hardware structures available divided between

Megafunctions: complex functions with a high level of abstraction (to be used later)

Others: particular functions related to a particular family of FPGAs

Primitives: Generic Functions

- to analyze the various graphic options available (for example that of being able to change the position of the gates while keeping the connections unchanged or vice versa to move the gates without the connections)

- give a mnemonic name to the various logic gates and especially to the input and output pins. PLEASE NOTE: use fairly specific names and avoid possible "reserved" words such as: "in", "out", "inout", "and", "inv", .... , as well as perhaps more unlikely words such as "for", "while", "goto" etc. etc.

Save the schematic and verify the correctness of the circuit through an analysis process.

Processing > Start > Start Analisys and Elaboration

Note that there are shortcuts both in the Tasks window and on the Top Bar

Correct any errors (double-clicking on the error message helps to highlight the affected part)

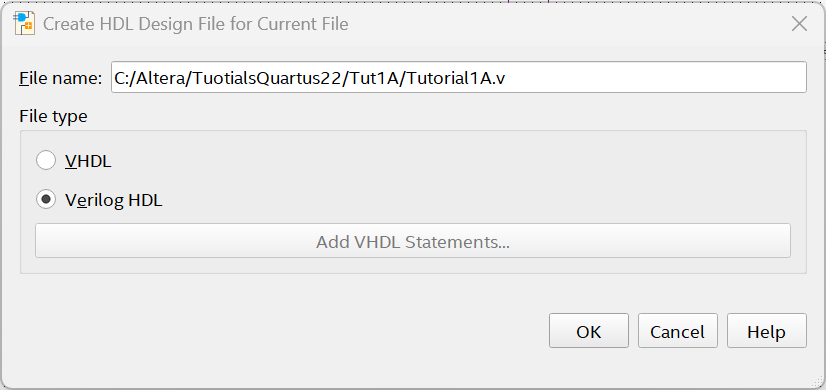
#### Functional simulation

Over the years, there have been different versions of simulators and methods for simulating the system to be analyzed. In this Tutorial we will see how to develop the simulation using a tool external to Quartus, i.e. Modelsim/Questa, but referring the reader to the specific tutorial related to the simulation with regard to the generation of stimulus signals and the visualization of the results.

**Preparation:**

Since the simulation tool uses systems described in textual form through a suitable HDL language (VerilogHDL or VHDL) the first operation is to transform the system generated into its equivalent HDL described form.

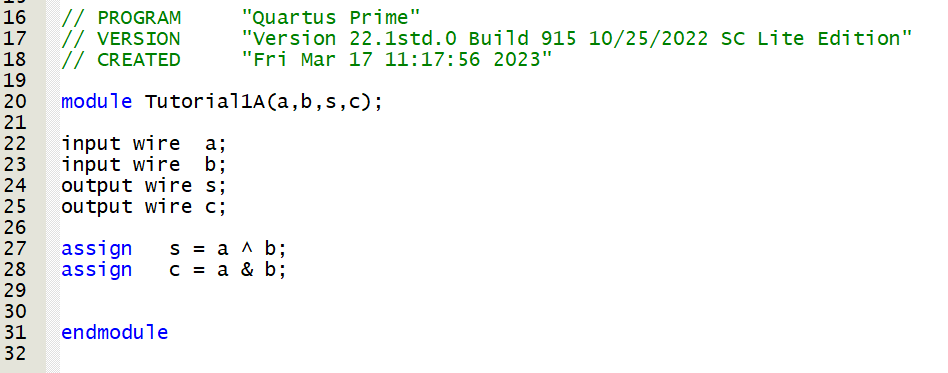
File > Create/Update > Create HDL Design File from current File



Choose a name for the output file

(OK)

Looking for the generated file in the folder you will find a file more or less similar to this:



Which describes the semi-adder developed in Verilog HDL (an alternative way to the schematic).

To avoid confusion at this point it is advisable to swap the two files in the project:   
include the verilog file and exclude the schematic.

Project > Add/Remove Files in Project

By clicking on the button with three dots, go to choose the verilog file generated just generated in the appropriate directory

(Open)

And at the same time remove the schematic (.bdf) from the list – highlight the file and  
  
 (Remove)



(OK)

In the Project Navigator Window (Files Mode): Right-click on the VerilogHDL file you just imported and

Set As TOP Level Entity (Ctrl – Shift – V)

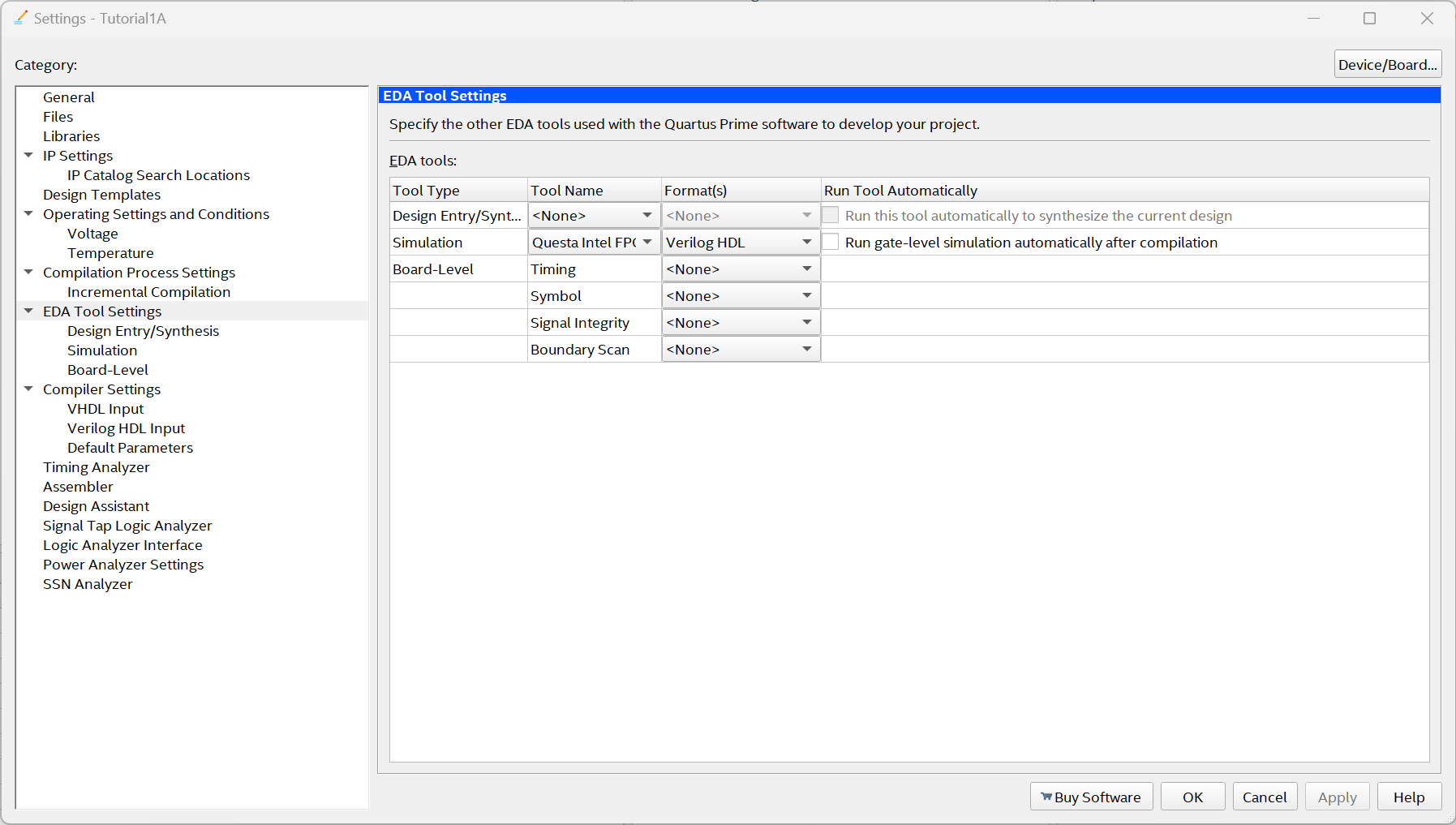
Relaunch "Analisys and Elaboration"

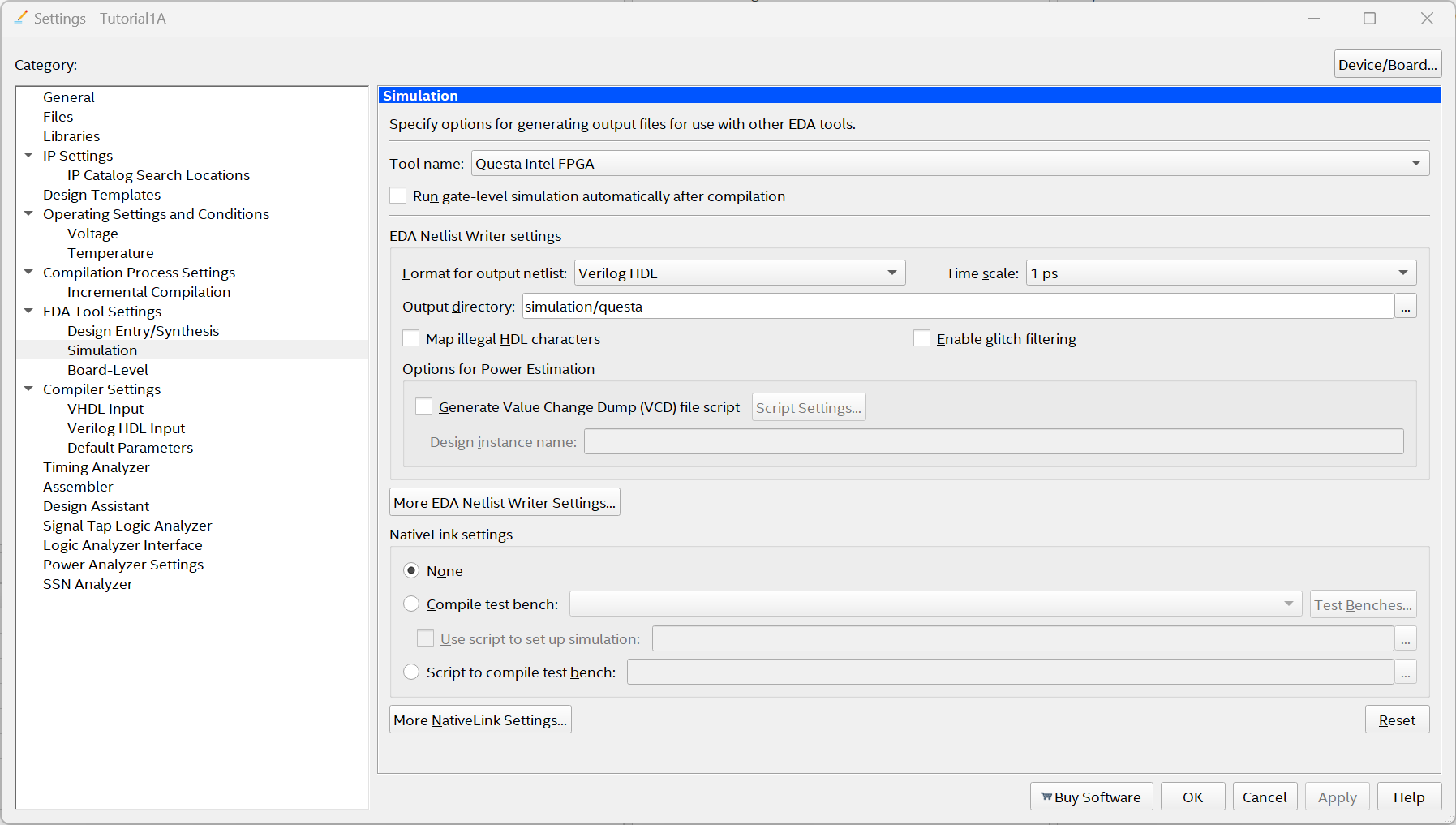
**Simulation**

At this point you can start the external simulator

Make sure you have defined the simulator to be used and its settings by accessing the various sub-menus

Assignment > Settings or (Ctrl – Shift – E )



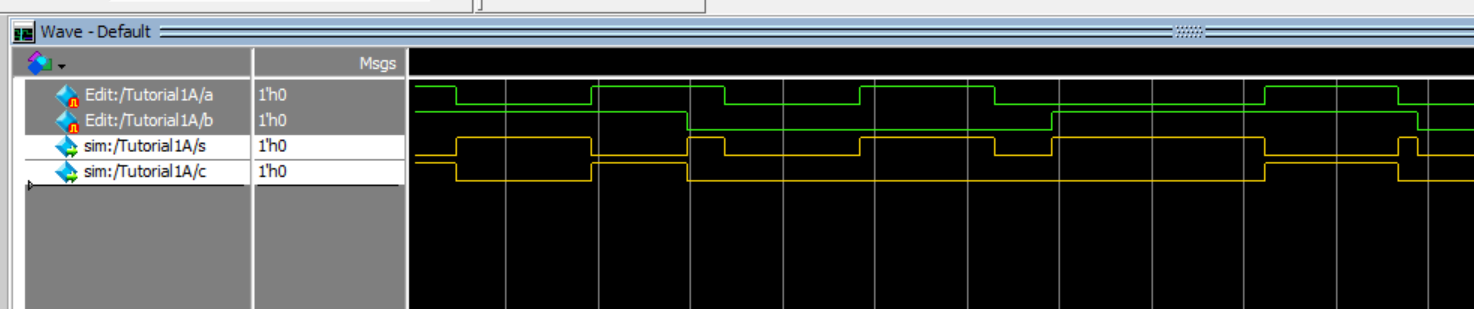


And launch the Simulation tool

Tools > Run Simulation Tool > RTL Simulation

Summarily:

1. In the Library Folder, open the "work" library
2. Inside find the file you want to simulate
3. Right Click – Create Wave
4. Simulate > Start Simulation
   1. Choose the file to simulate in the work library (TOP Level Entity)
   2. Click on the “Optimization Options” button and if necessary choose "Apply Full visibility ..."
5. In the Wave window, include the signals to be displayed (by dragging them from the objects folder) and delete the signals of stimuli that are NOT configured
6. Right click on the stimulus signals   
   Edit > Wave Editor > Create / Modify Waveform
7. Define the desired signals
8. To edit signals using the   
   Wave mouse > Mouse mode > Edit mode
9. Change the stimulus signals at will
10. Launch the   
    simulation Simulate > Run > Run -all
11. If necessary, change the properties of the waveforms to make the results more readable



If the simulation provides correct results, you can proceed

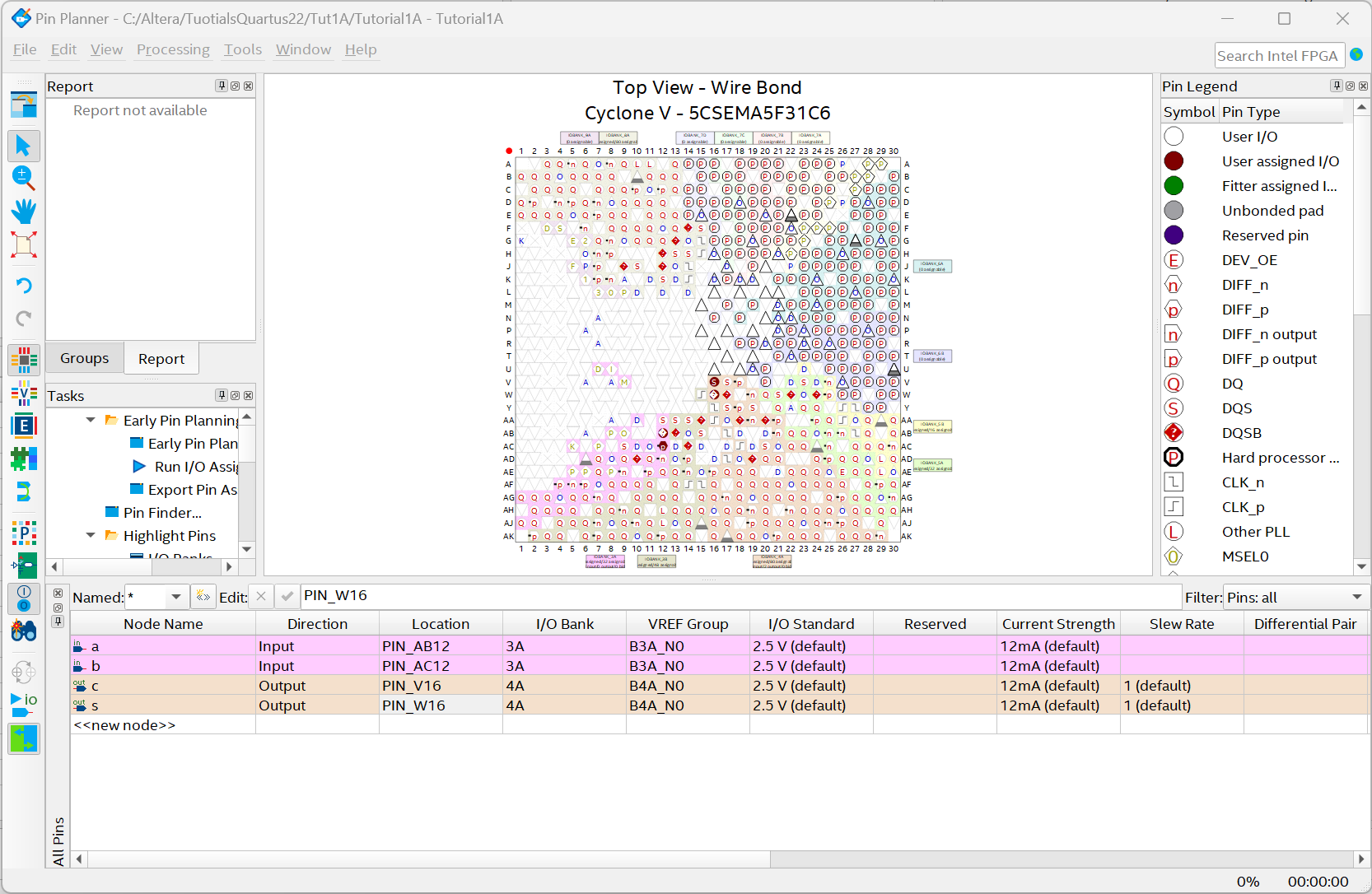
#### Defining constraints [8-9]

At the moment we will consider just how to set position constraints for the input and output pins, in order to ensure that the inputs are associated respectively with switches 0 and 1 and that the outputs are connected to the red LED number 0 and 1:

The DE1-SoC user manual [] shows that SW[0] is connected to pin **PIN\_AB12**, while SW[1] is connected to pin **PIN\_AC12**. In addition, the LEDs are connected to the **PIN\_V16** and **PIN\_W16**. It is therefore mandatory to associate these positions to the input and output pins of our circuit (try other associations if necessary, but always with the constraint that obviously the switches can only be used as inputs. Although both the board and the FPGA have protections, from a general point of view, assigning a logic value to a signal that can be externally forced to a different logical value creates a conflict that could **damage** the device itself.

Assignments > Pin Planner (ctrl-shift-N)

Edit the table as shown in the figure:



Close the window

Note1: a similar result can also be obtained using the "assignment editor" tool which can also be used to assign constraints of other kinds.

Assignments > Assignments Editor (ctrl-Shift-A)

Note2: Another more convenient way to assign pinout constraints (especially useful when the number of pins to be assigned is quite high is to use "System Builder": a tool made available by the manufacturer of the Board that in graphic form and based on the devices on the board you want to adopt, proposes a skeleton of a project to be modified as desired, but already accompanied by all the useful constraints. This tool will be used in future tutorials.

At this point you can proceed with the complete compilation of the project. This involves several steps "Analysis", "Synthesis", Place & Route", "Generate programming file", "Time Quest Timing analysis", etc. etc., which are automatically carried out in sequence:

Processing > Start Compilation (ctrl-L).

#### Time simulation:

At this point the circuit is completely realized and perfectly defined in all its internal structure of the FPGA, so a further simulation could be carried out at the lowest level of abstraction that highlights all the delays of the various portions of the circuit.

Tools > Run Simulation Tools > Gate Level Simulation

It should be noted that within the directory dedicated to simulation files there is now a "gate work" directory

<project name>\simulation\questa\gate\_work

Ready for simulation at the gate level, and in the upper directory the file

<filename>.vo

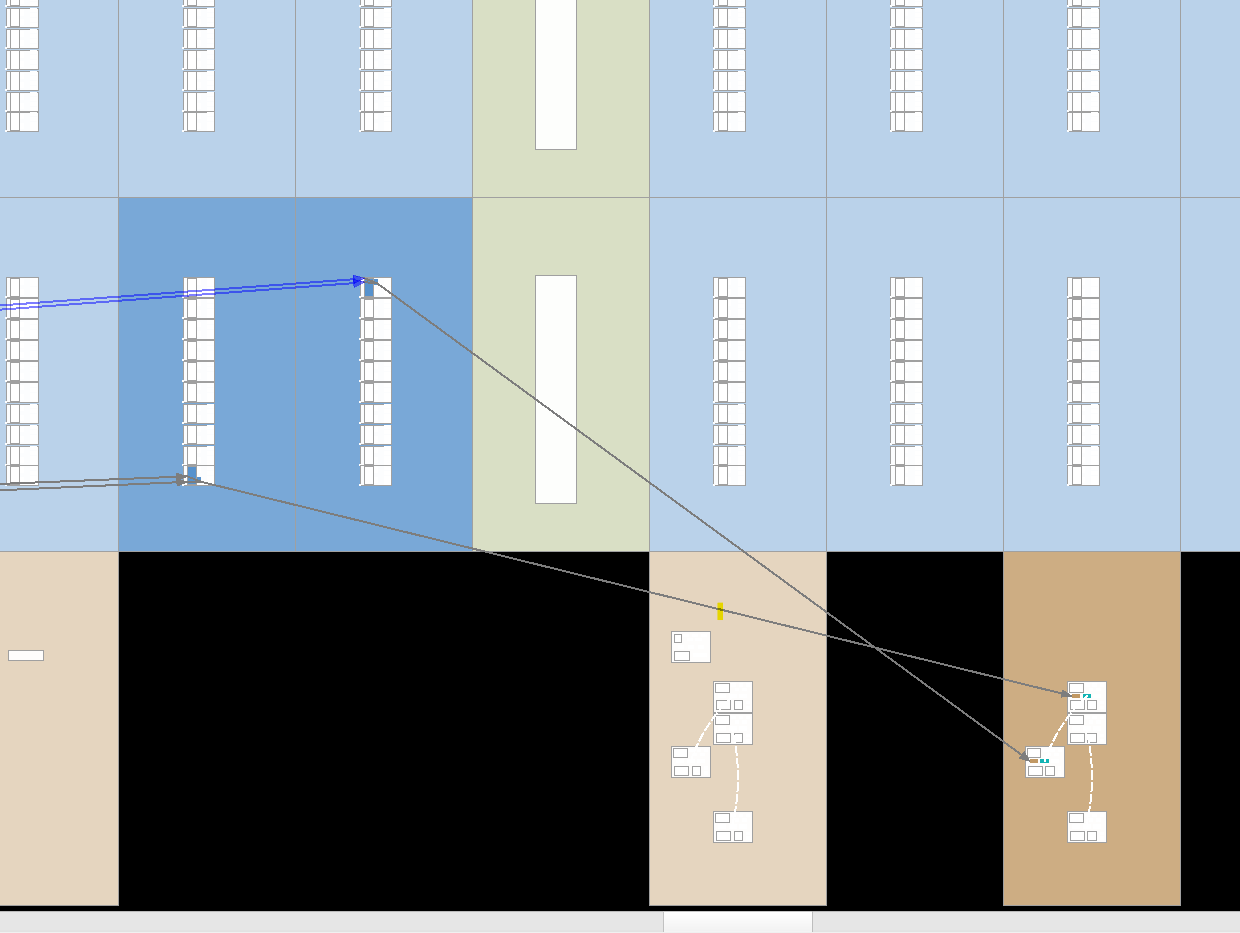
Which describes, at the structural level, all the functional blocks and I/O ports that make up the final circuit at the gate level.

Unfortunately, however, for the Cyclone V Family, the manufacturer  **has NOT** provided the libraries that define the physical characteristics of the various cells and individual devices that make up the FPGA, instead providing the "Timing Analyzer" as a tool for timing analysis that will be analyzed in a subsequent tutorial.

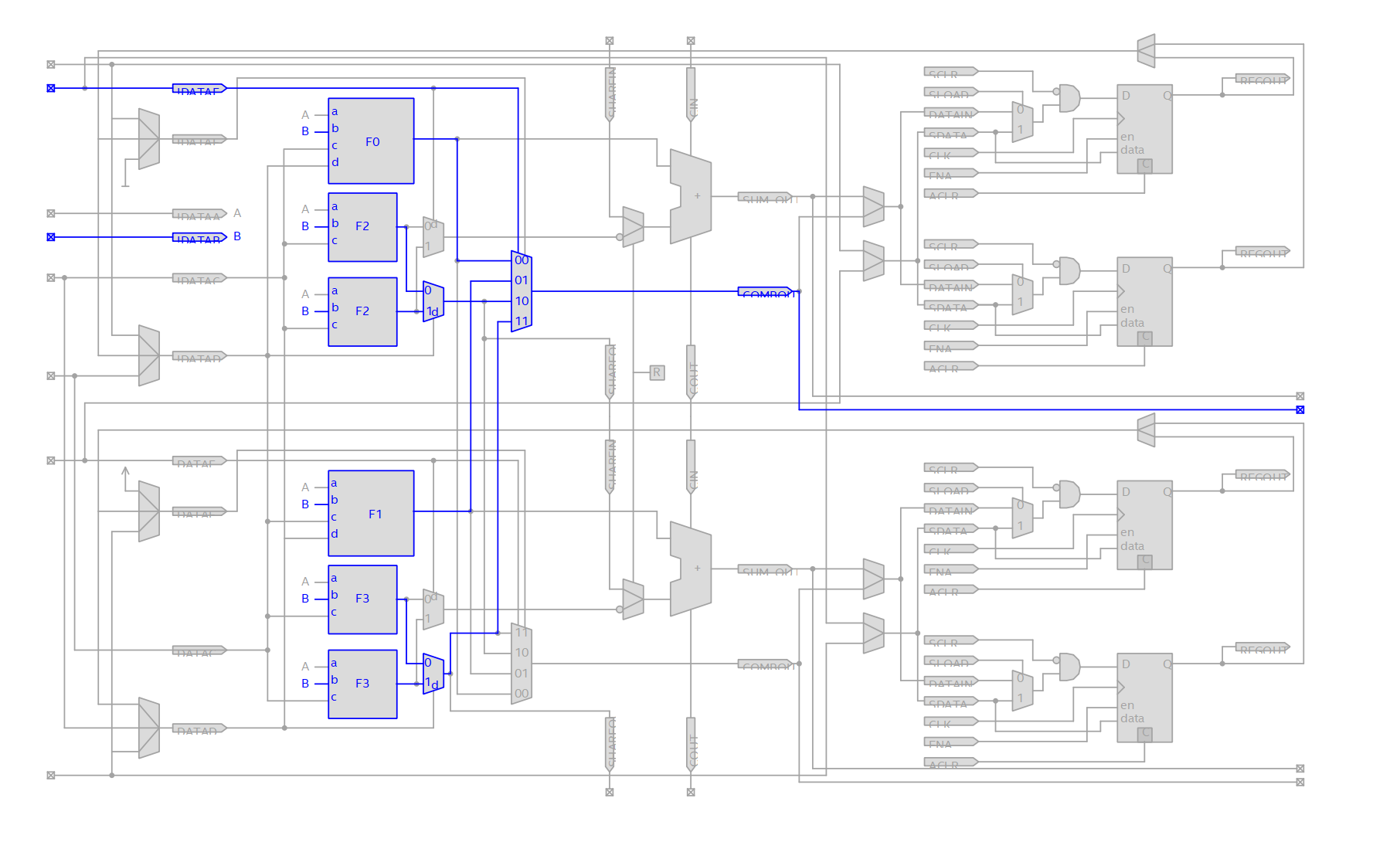
#### Circuit layout display [10]:

Tools > Chip Planners

Use the zoom to highlight small parts of the circuit and by double-clicking on the single cell highlight its specific configuration.



Highlight in particular the LEs involved in the project, their connections, the I/O ports involved, the connections and their internal configurations

This tool also allows you to change the position within the FPGA of the various LEs used, for example to reorganize the connections and consequently modify the delays. But on circuits of considerable complexity it is good to delegate this function to the automatic synthesis system.

#### Visualization and analysis of the various process reports:

Most likely the window containing all process reports should already be open, otherwise it can be called up with

Processing > Compilation Reports (ctrl-R)

The various reports are organized by sections as are the various stages of the compilation process. Analyze them in detail, in particular focus on the resources employed within the FPGA

Analysis & Settings > Resources Usage Summary

on the congruence of the pinout

Fitter > Pin-out File

#### Detailed analysis of delays

To carry out a detailed analysis of the propagation delays of the various signal paths, a dedicated tool can be used for this purpose, called "Timing Analyzer" []

Tools > Timing Analyzer

In the new window, create the timing information

Netlist > Create Timing Netlist

And choose the analysis mode (**Post-fit**) and the model to be adopted (**Slow**).

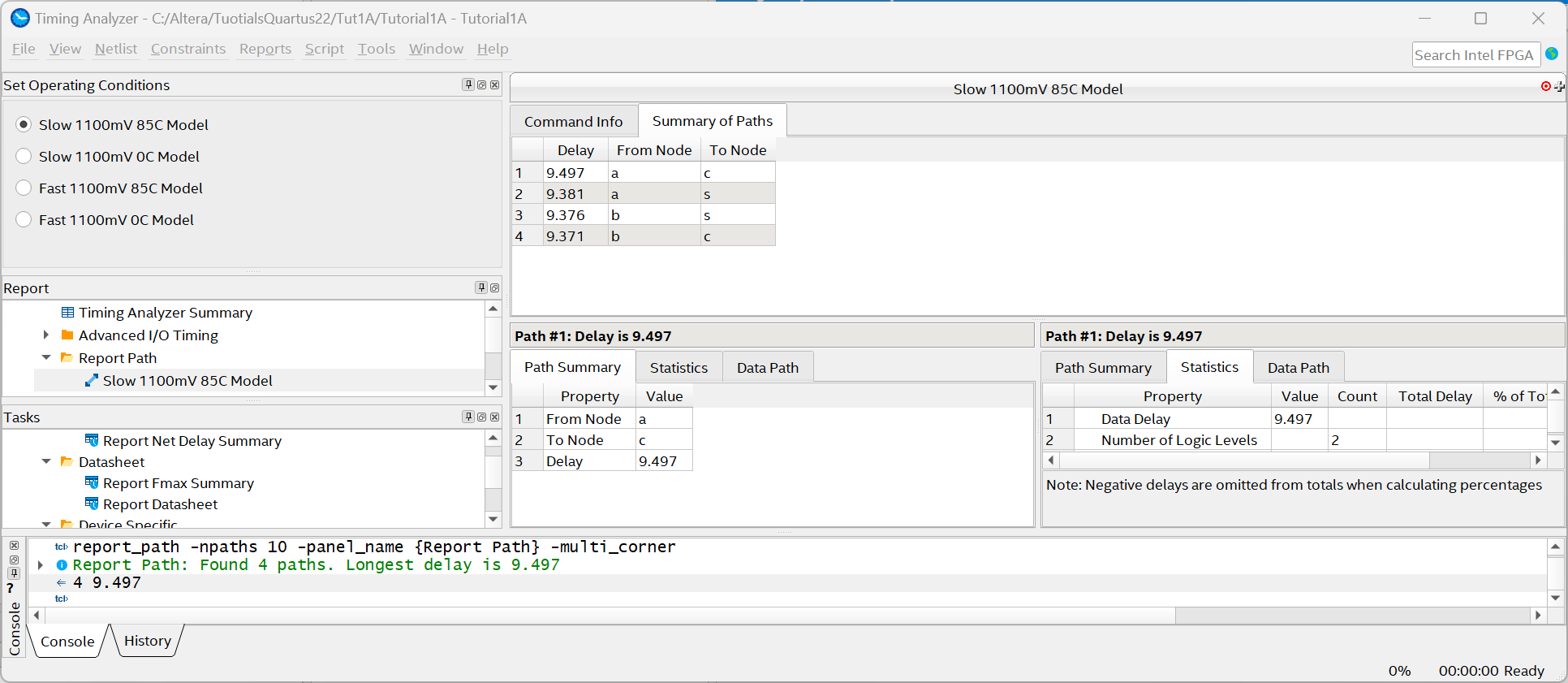
At this point it would be necessary to fix the clock of the system, its working frequency and the constraints within which the system should be subject, but given the simplicity of the project in question we can move on to the next phase:

Netlist > Update Timing Netlist

Now view the report of interest

Reports > Custom Reports > Reports Path

The window that opens has the function of a filter to display, within a complex project, only the paths of our interest, but in the case in question you can view all the paths as they are only four. Therefore, set the variable "**Reports Number of Paths**" to a value higher than 4 and click on "**Report Paths**"

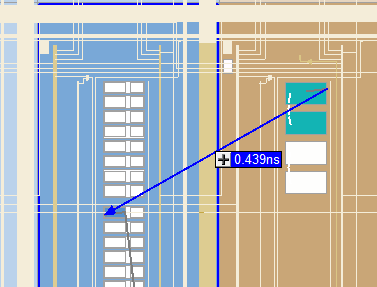


In the window that opens, you will have a detailed report in all the steps of the four paths that make up our circuit, from the input of the signal to the two outputs of the latter. Furthermore, by right-clicking on the path of our interest and choosing "**locate path**" you can identify the path in question in some of the possible visualizations of the circuit.

For example, by displaying them within the "Chip Planner", and using the internal command

View > Show Delays

You can view the delays of the various steps of the route.



It should also be noted that the "Timing Analyzer" tool is not only a control tool, but can be used to set constraints to be passed to the synthesizer, so that during the construction of the circuit it can verify if these constraints are respected or, if not, try other solutions.

#### On-board testing

The ultimate step is to download the bitstream (which contains all the information on the FPGA configuration) on the FPGA itself

* Verify that the MSEL micro-switches on the back of the board are in   
  MSEL configuration [1:6]= 6'b01001-
* Connect the USB download cable
* Turn on the card

After a moment, the board will be configured according to the configuration file residing in the flash, which by default provides for the creation of graphic patterns on the LEDs.

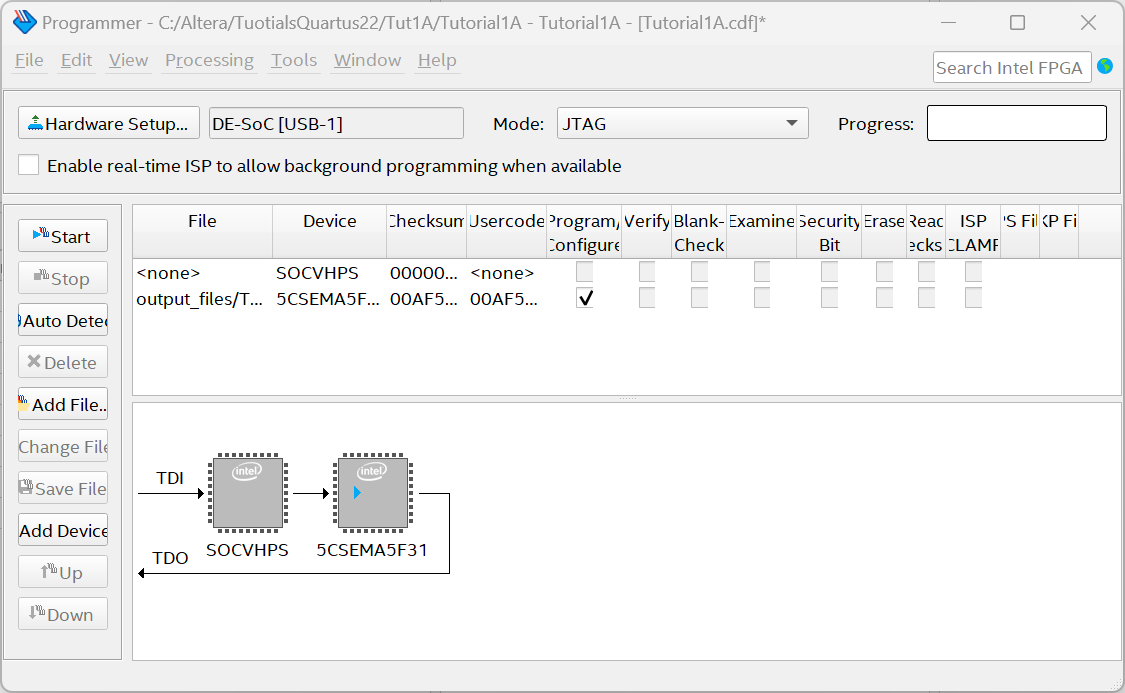
Tools > Programmer

If necessary - Click on "Hardware setup" and choose DE-SoC

Possibly – Auto Detect and choose 5CSEMA5

Add File – in the "output\_files" folder choose "<name>.sof

The programming chain must consist of two serialized elements (SOCVHPS and 5SEMA5F31) . The element 5SEMA5F31 should contain the programming files. If there were others elements, they must be removed



Click on Start

Verify that the system is working properly using swithes on the board and looking at the leds.

## Bibliography

[1] Terasic : "**DE1-SoC User Manual – Terasic**" *https://www.terasic.com.tw/cgi-bin/page/archive\_download.pl?Language=China&No=836&FID=ae336c1d5103cac046279ed1568a8bc3*

[2] Altera Corporation: **"Cyclone II FPGA Starter Development Board -Reference Manual**" *<http://www.altera.com/literature/manual/mnl_cii_starter_board_rm.pdf>*

[3] Altera Corporation: **"Quartus II Handbook Version 13.1 - Volume 1: Design and Synthesis**"

*<http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf>*

[4] Altera Corporation: "**Cyclone II Device Handbook**"   
<http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf>

[5] Altera Corporation: "**Documentation on Cyclone II Devices**"  
<http://www.altera.com/literature/lit-cyc2.jsp>

[6] Altera Corporation: "**My First FPGA Design Tutorial**" *<http://www.altera.com/literature/tt/tt_my_first_fpga.pdf>*

[7] Altera Corporation: "**Quartus II Introduction Using Schematic Designs**" *<ftp://ftp.altera.com/up/pub/Altera_Material/13.0/Tutorials/Schematic/Quartus_II_Introduction.pdf>*

[8] Altera Corporation: "**Quartus II Handbook Version 13.1-Volume 3: Verification**"  
<http://www.altera.com/literature/hb/qts/qts_qii5v3.pdf>

[9] Altera Corporation: "**Using the Assignment Editor in the Quartus II Software**"  
<http://www.altera.co.jp/literature/wp/wp_asigned.pdf>

[10] Altera Corporation: "**Constraining Designs**"  
<http://www.altera.com/literature/hb/qts/qts_qii52001.pdf>

[11] Altera Corporation: "**Analyzing and Optimizing the Desig Floorplan with the Chip Planner**" *<http://www.altera.com/literature/hb/qts/qts_qii52006.pdf>*

[12] Altera Corporation: "**TimeQuest Timing Analyzer - Quick Start Tutorial**"  
*<http://www.altera.com/literature/ug/ug_tq_tutorial.pdf>*

[13] Altera Corporation: "**Quartus II TimeQuest Timing Analyzer Cookbook**"   
*<http://www.altera.com/literature/manual/mnl_timequest_cookbook.pdf>*

[14] Altera Corporation: "**The Quartus II TimeQuest Timing Analyzer**"   
*<http://www.altera.com/literature/hb/qts/qts_qii53018.pdf>*[15] Altera Corporation: "**Quartus II Introduction Using Verilog Design**" *<ftp://ftp.altera.com/up/pub/Tutorials/DE2/Digital_Logic/tut_quartus_intro_verilog.pdf>*

[16] Stuart Sutherland: "**Verilog® HDL - Quick Reference Guide**"   
*<http://www.sutherland-hdl.com/online_verilog_ref_guide/verilog_2001_ref_guide.pdf>*

[17] Synopsys Corporation: "**FPGA Compiler II / FPGA Express Verilog HDL Reference Manual**"  
*<http://classes.soe.ucsc.edu/cmpe225/Fall01/synver.pdf>*

[18] "**THE VERILOG HDL LANGUAGE FOR SYNTHESIS AND THE. SIMULATION OF DIGITAL CIRCUITS** http://unina.stidue.net/Sistemi%20Elettronici%20Programmabili/Materiale/Manuale%20Verilog.pdf

[19] Carlo Brandolese - Politecnico di Milano: "**Introduction to the VHDL language - Theoretical aspects and design examples",** *[http://www.fabiopanozzo.it/download/Reti\_logiche/VHDL%20(Lecture notes%20a%20cura%20di%20Carlo%20Brandolese).pdf](http://www.fabiopanozzo.it/download/Reti_logiche/VHDL%20(Dispensa%20a%20cura%20di%20Carlo%20Brandolese).pdf)*

**[20] FPGAcademy** [**https://fpgacademy.org/tools.html**](https://fpgacademy.org/tools.html)