Design Flow

Know your tools

Rui Machado
Embedded Systems Research Group (ESRG)
Centro ALGORITMI, Universidade do Minho

https://esrg.algoritmi.uminho.pt/

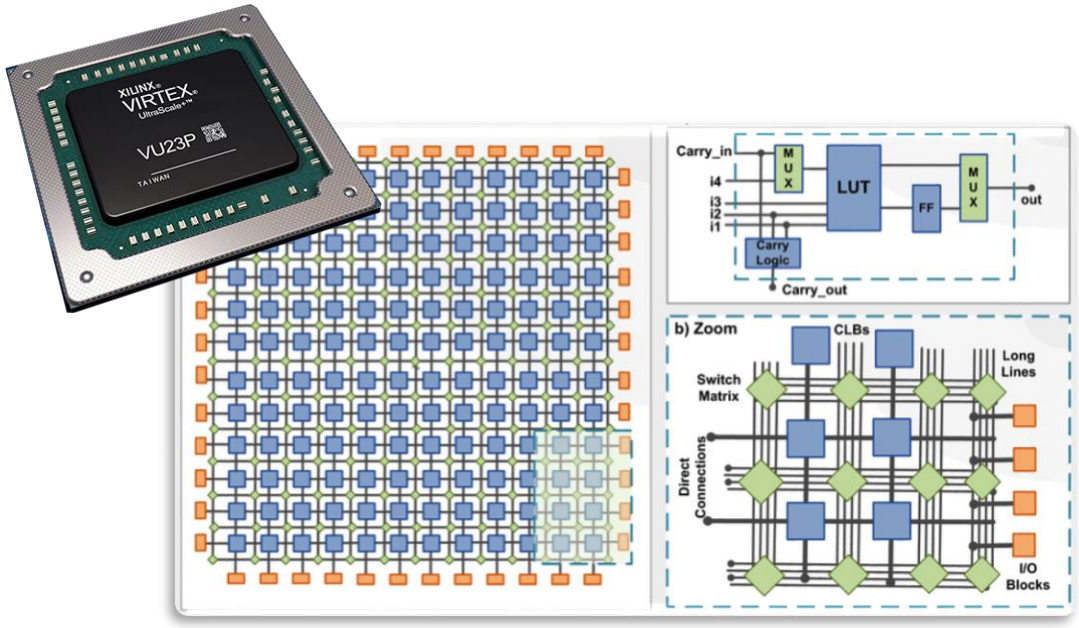
CENTROALGORITMI

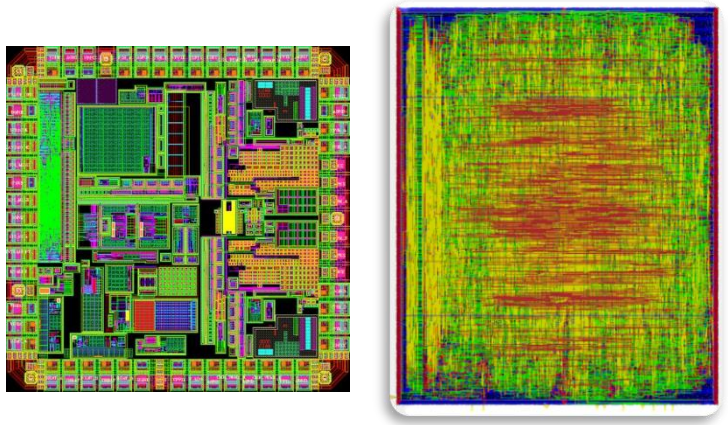
FCT Fundação para a Ciência e a Tecnologia

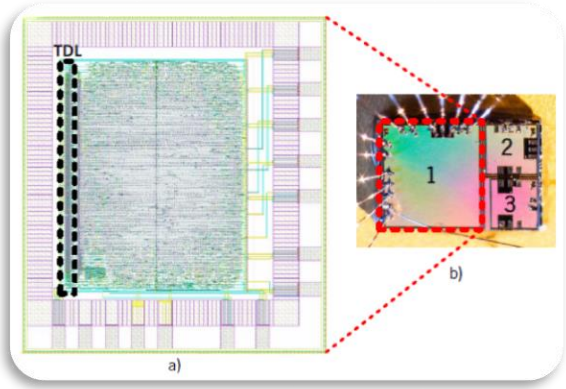
TABLE OF CONTENTS

Design Flow

FPGA Vs ASIC

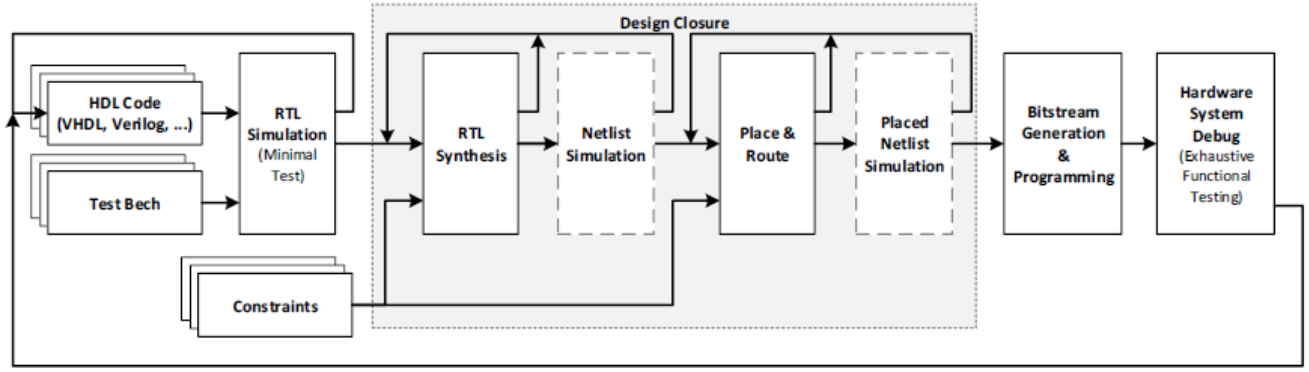


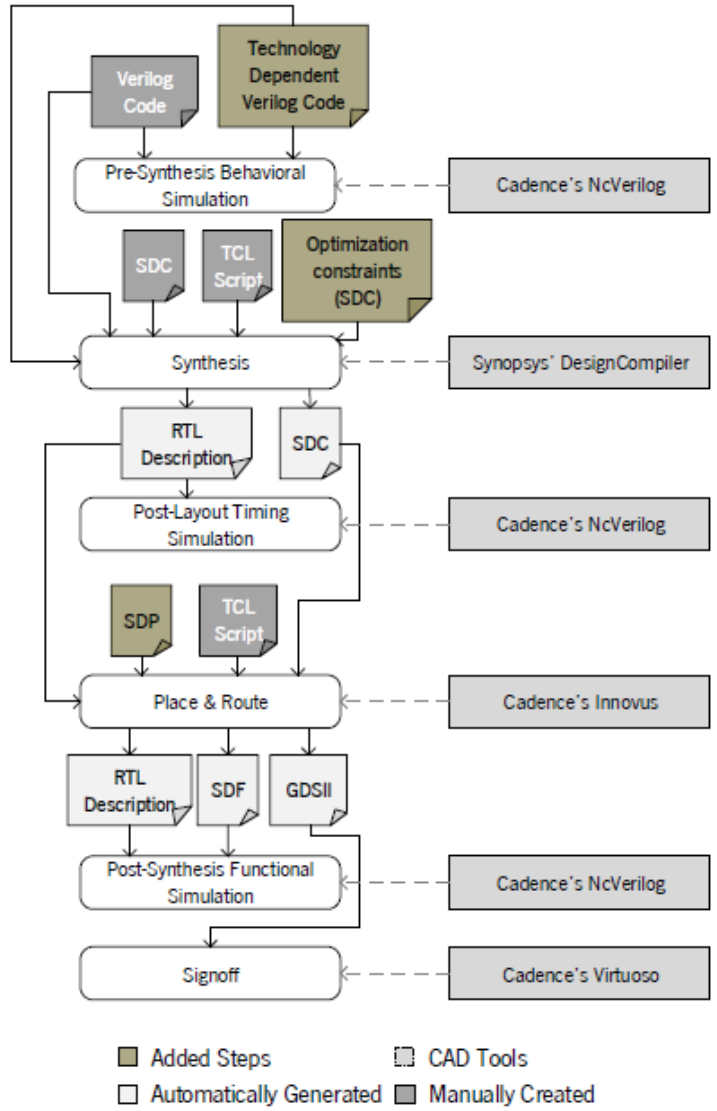




Design Flow

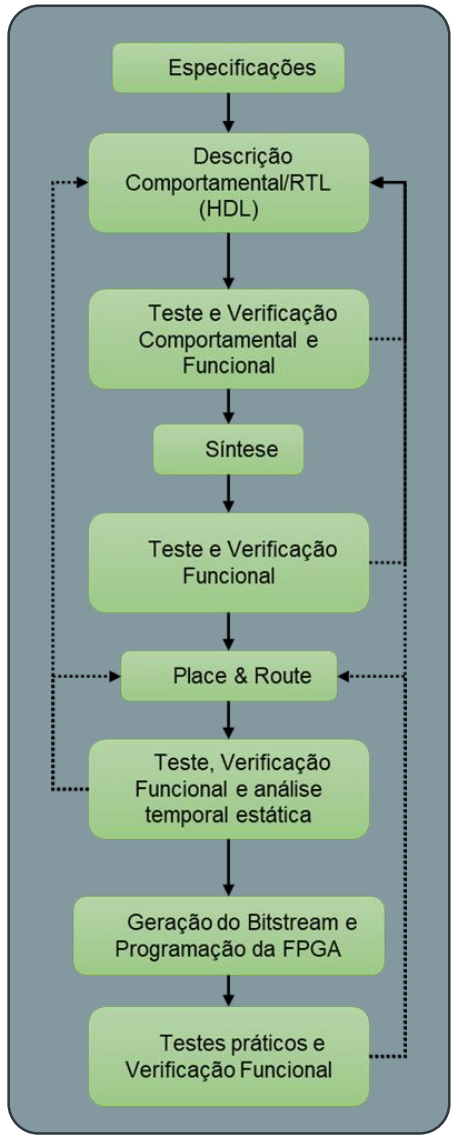
- FPGA Vs ASIC





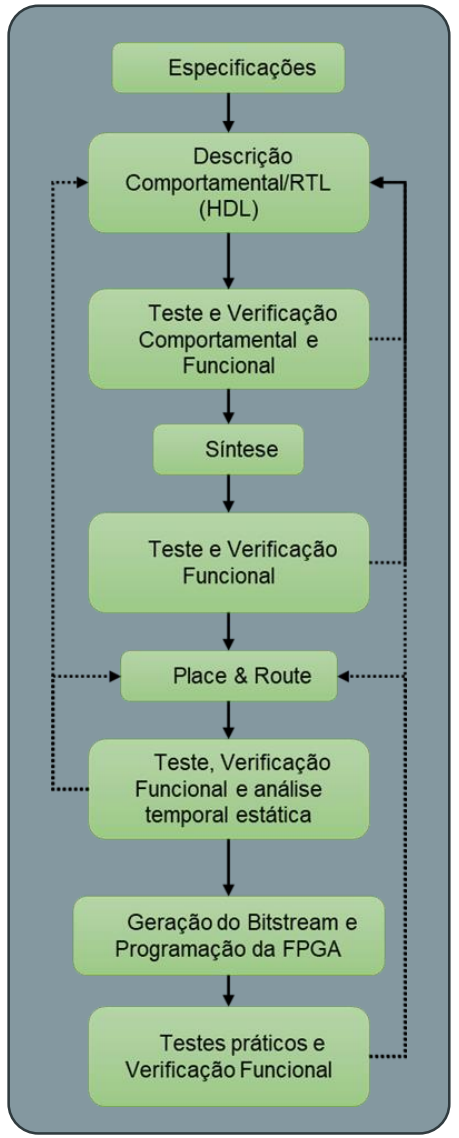
Design Flow

- FPGA
  - First step consists into creating the HDL files that will answer the identified requirements for the system to be developed
  - This should be done using a modular approach and the RTL for each module should be checked to guarantee that the HDL developed is, in fact, generating the expected hardware (as a designer you should always have an idea of the RTL that you should be generating, don't let the tools do whatever they want!)
  - Testbench files should also be created to test the developed modules. These test should be independent of the implementation and should be possible to reuse then throughout the multiple verification phases of the design flow



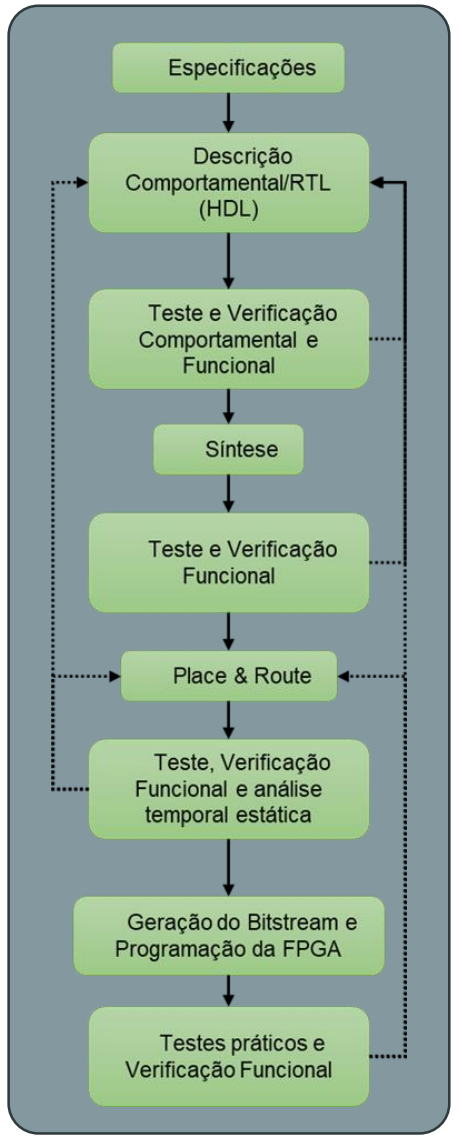
Design Flow

- FPGA
  - Next step is to do behavioral simulations using the testbenches created
    - Testbenches should be self-tested
  - If the tests fail, we iterate back to the design phase to correct the errors
  - If the tests succeed, you move forward to synthesis
  - In synthesis the RTL is mapped to a specific technology, however, there is still no information regarding where the logic will be placed nor which routing paths will be used.
  - Some code might not be synthesizable, so be careful as these usually result in a synthesized design that is different from the one you were expecting



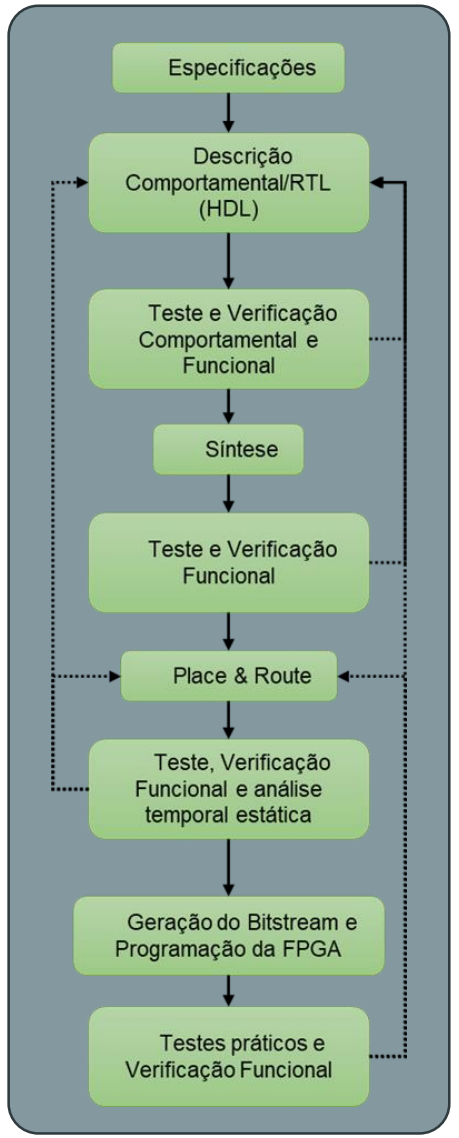
Design Flow

- FPGA
  - Implementation phase places the logic and selects the routing paths to connect the logic according to the designed RTL.
  - It also performs a set of design optimizations once again (exploring different placements and routing paths to help the design meet timing, area and power constraints).
  - Usually, the implementation phase in FPGA requires minimal to none intervention. The tools are usually very capable and automate most steps. In very specific and/or complex designs however, it might be needed to manually route some paths or manually place some logic to comply with the design constraints



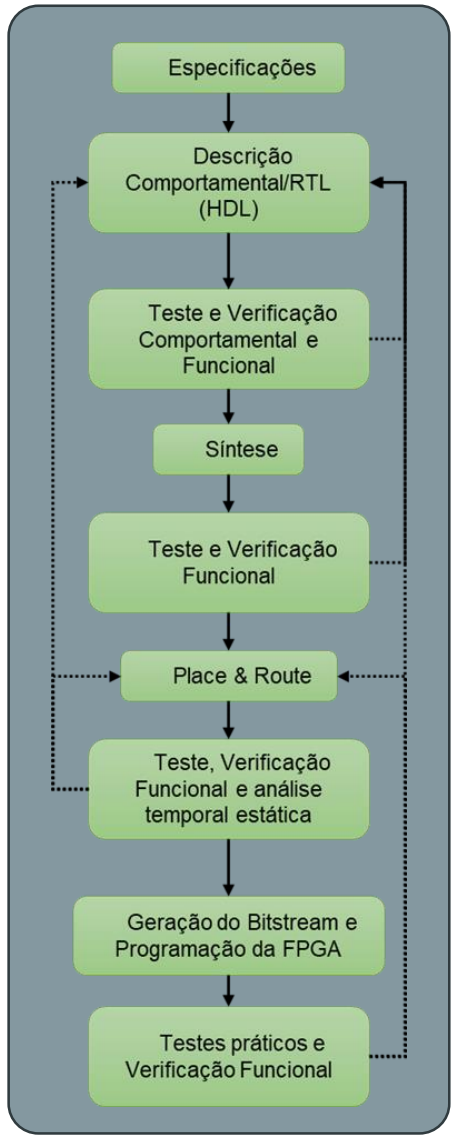
Design Flow

- FPGA
  - After implementation it is now possible to obtain detailed information regarding the system timing, such as, propagation delays.
  - Again, after checking the implementation netlist (that should be similar to the synthesized one), it is time to do functional verification.
  - If the design passes verification we can move to bitstream generation. If not, we might go back to the design phase or change the design placement and routing and re-run implementation.



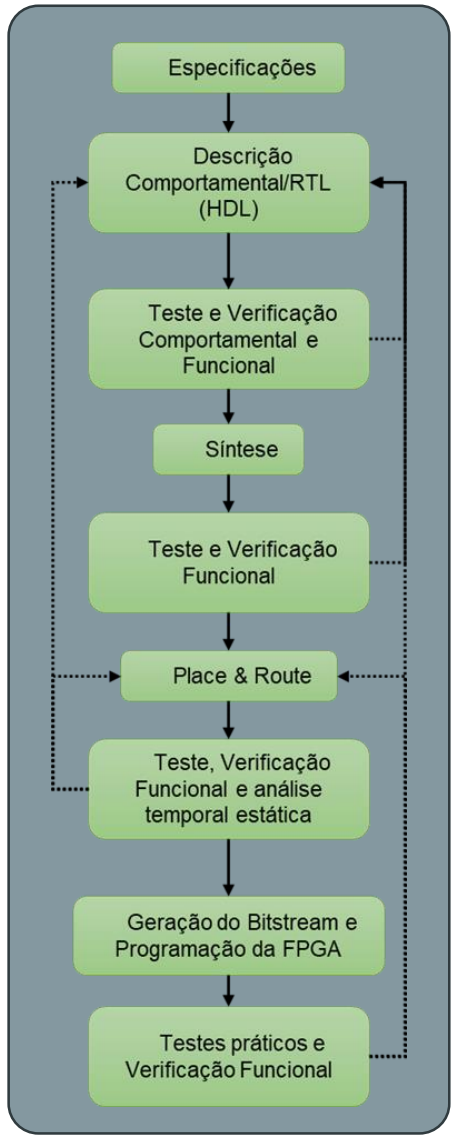
Design Flow

- FPGA
  - Depending on the design, timing simulation might also be done. These are time consuming simulations. Most of the time, if the system passes the static timing analysis (STA), everything is ok, and no timing simulation is needed. Nevertheless, it is possible to be done
  - STA done automatically by the tools as long as a clock constraint exists.
  - It guarantees that setup and hold times are met
  - Since in FPGA the available resources are already placed and the possible routing paths are already defined, when a design fails STA, is usually due to tight clock constraints and not due to poor placement and routing



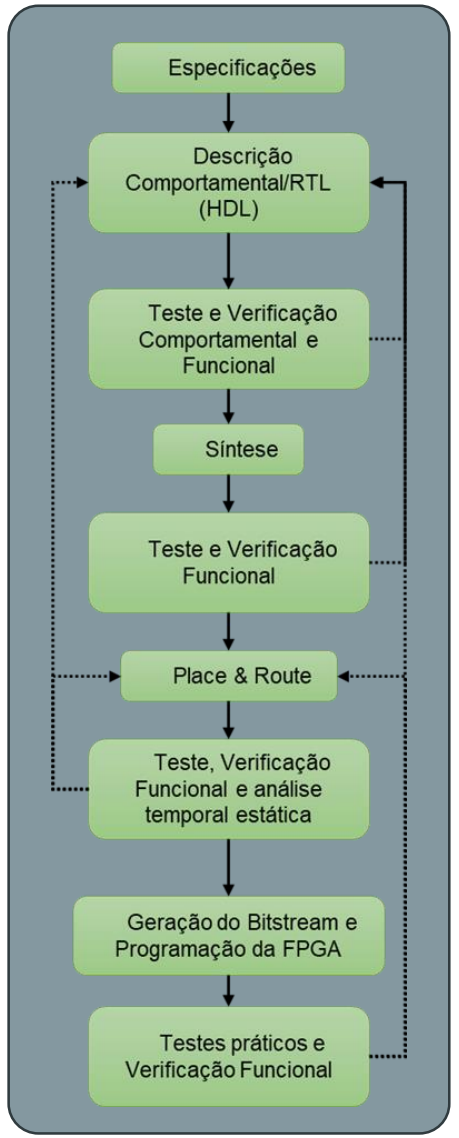
Design Flow

- FPGA
  - If the design complies with all the constraints and passes all the tests the bitstream can be generated
  - The bitstream is a file responsible for configuring the FPGA logic to behave as defined by our design
  - During bitstream generation DRC is performed
  - Once we have the bitstream we can program the FPGA and test the design in the physical device
  - Although not common (at least not if you properly test the design), it is possible for a design to not work on the physical device, even if all the tests were successful



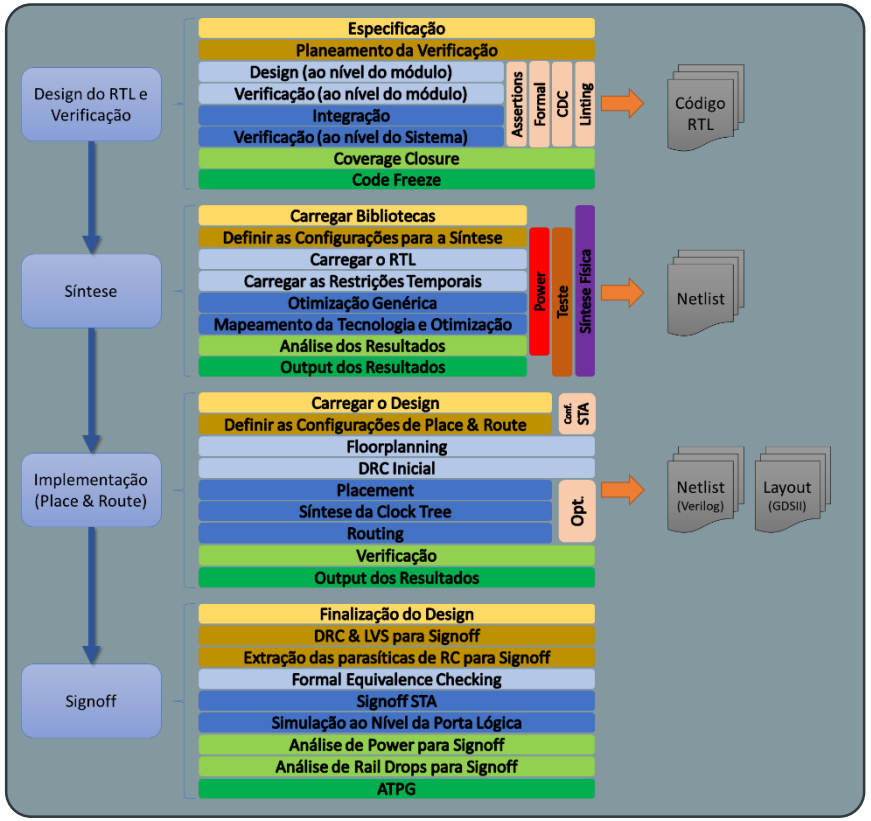
Design Flow

- FPGA
  - In these situations, logical analysers can be included in the design to help debug the problem
  - One can also try to replicate the physical inputs battery into a testbench and perform timing simulations to try to replicate the error
  - When these errors occur, it is important to try to understand if they are random or systematic and the input sequence that is driving the system to an erroneous behavior



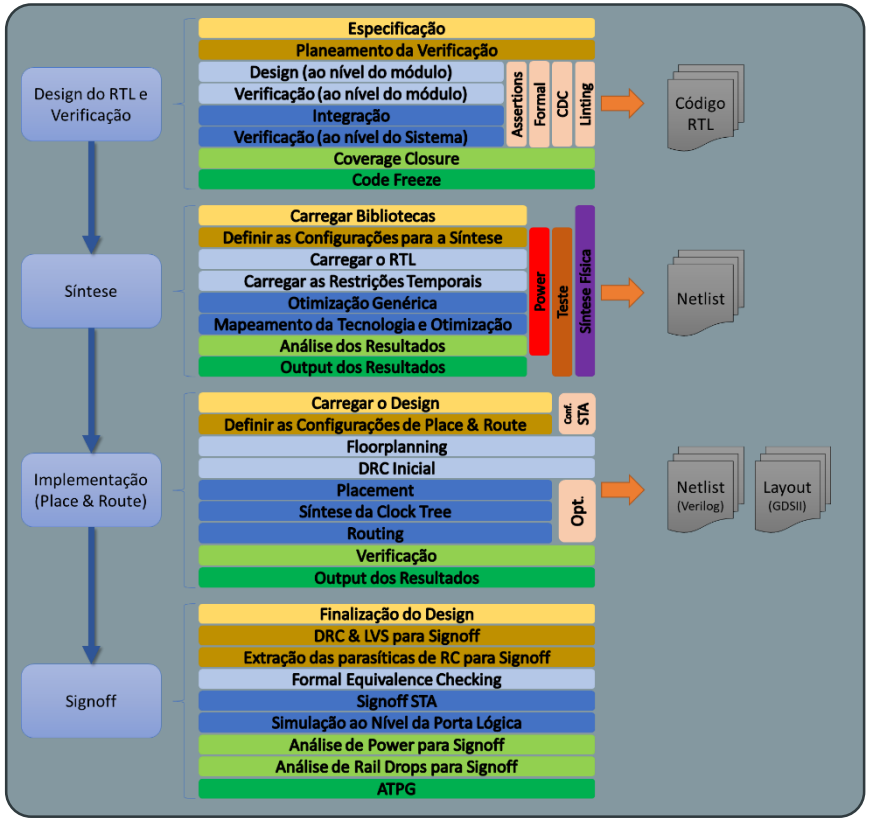
Design Flow

- ASIC
  - Verification assumes a much important role since after fabrication no change is possible on the hardware. This includes testbenches as well as STA, timing simulations, DRC and LVS, etc.
  - Place & Route requires a much bigger effort from the designer with few automated tasks.
  - The clock tree must be created and verified by the designer
  - Floorplan is done manually as well as power planning



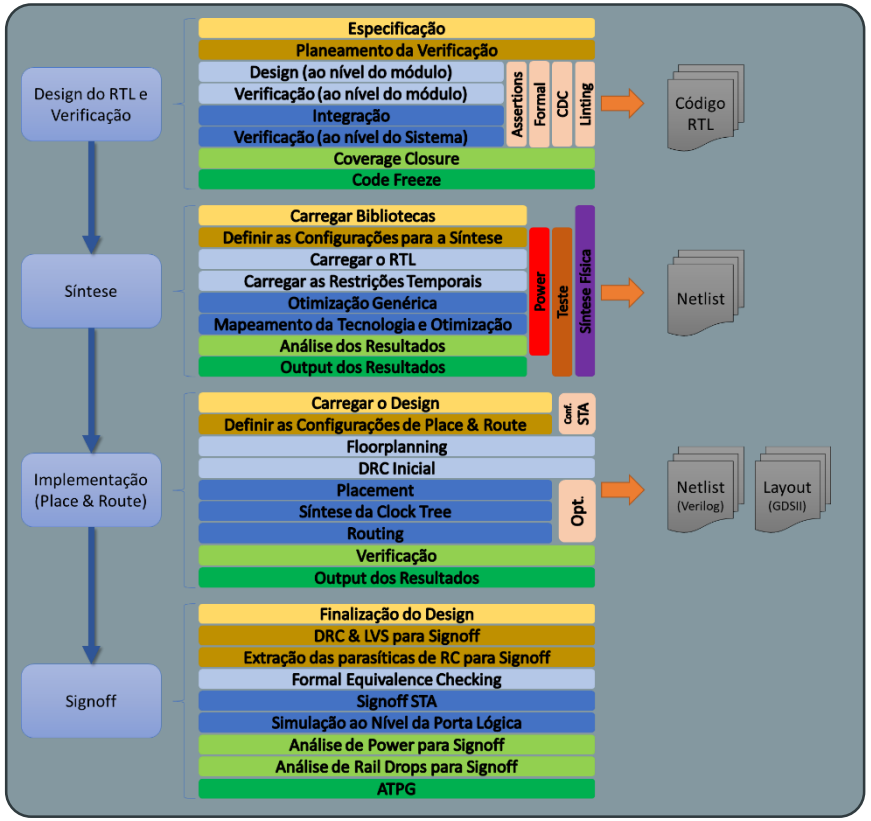
Design Flow

- ASIC
  - The process starts in the same way as in FPGA (identify requirements, address the design hierarchically, define a test battery, and development of the RTL code)
  - As in FPGA, all the processes involved in the design flow can be performed using a GUI or using TCL scripts
  - Synthesis requires the designer to first define the technology libraries that he intend to use, as well as the desired optimizations among others



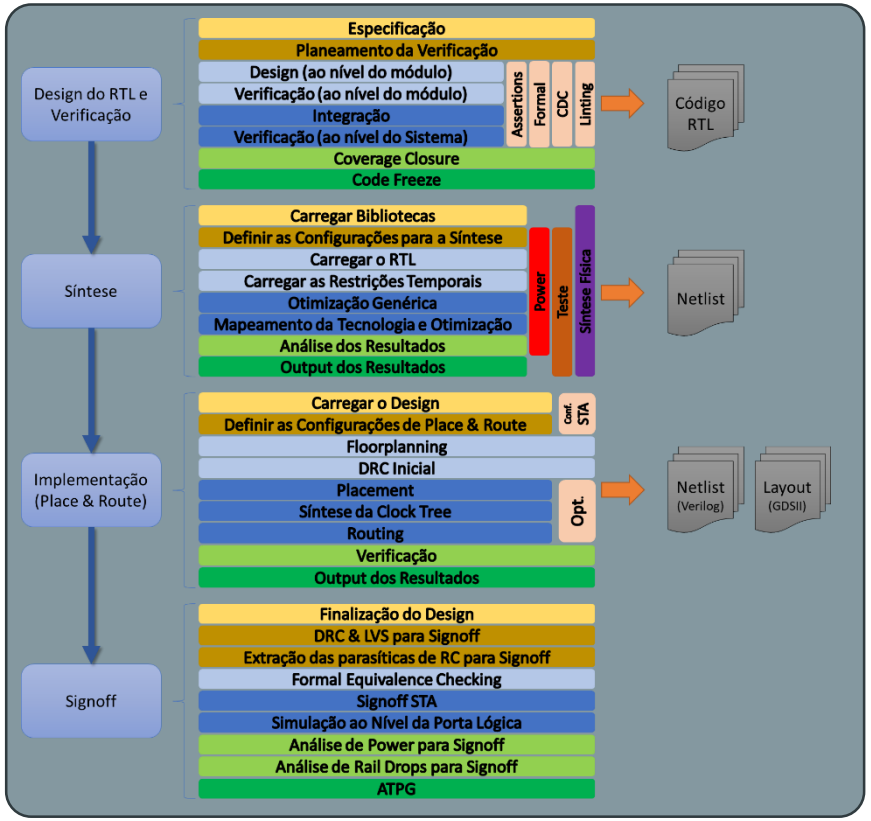
Design Flow

- ASIC
  - After configuring the tools, the next step is to read the RTL files, analyze them and elaborate the design (create the design's technology independent Netlist)
  - The elaboration step includes a technology independent optimization step as well.
  - The next step is to generate a technology dependent Netlist (compile the design). This compilation step also includes a technology dependent optimization.



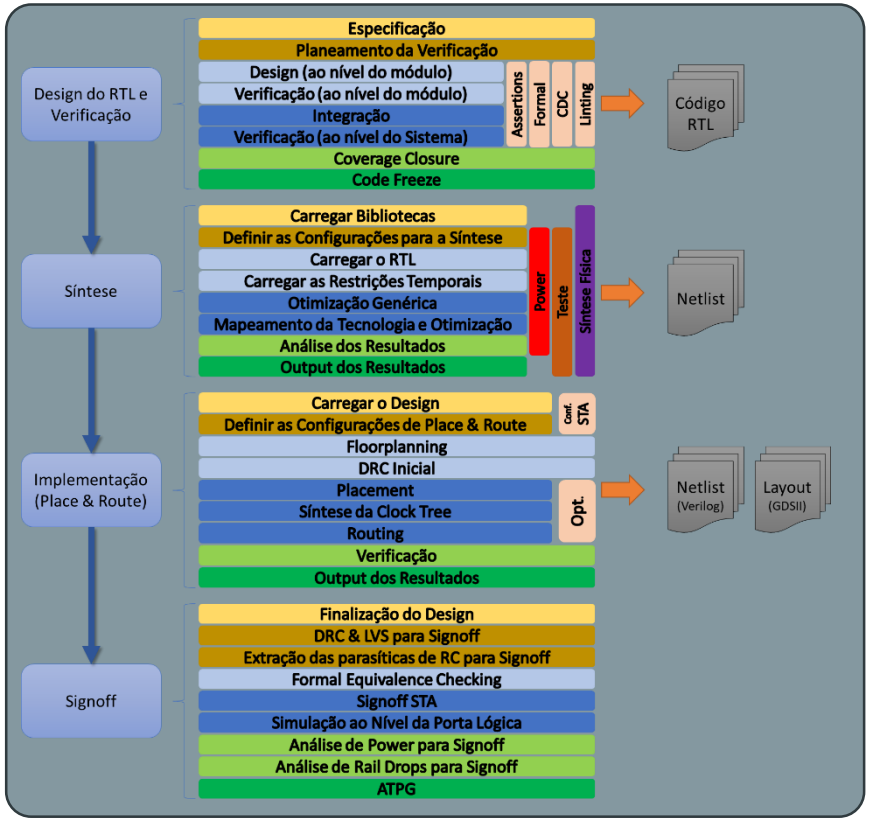
Design Flow

- ASIC
  - This optimization process may include Datapath and logic gates restructure, logic remapping, logic element upsizing or downsizing (allows for fanout and rise and fall times adjustments)
  - The synthesized Netlist is then imported to the place & Route tool, as well as the files responsible for describing the physical properties of the technology library in use, together with the timing constraints for the design (for fast and slow corners), technology parasitic information, number of layers available (for example TSMC 180nm has 6 layers available), among other configuration options



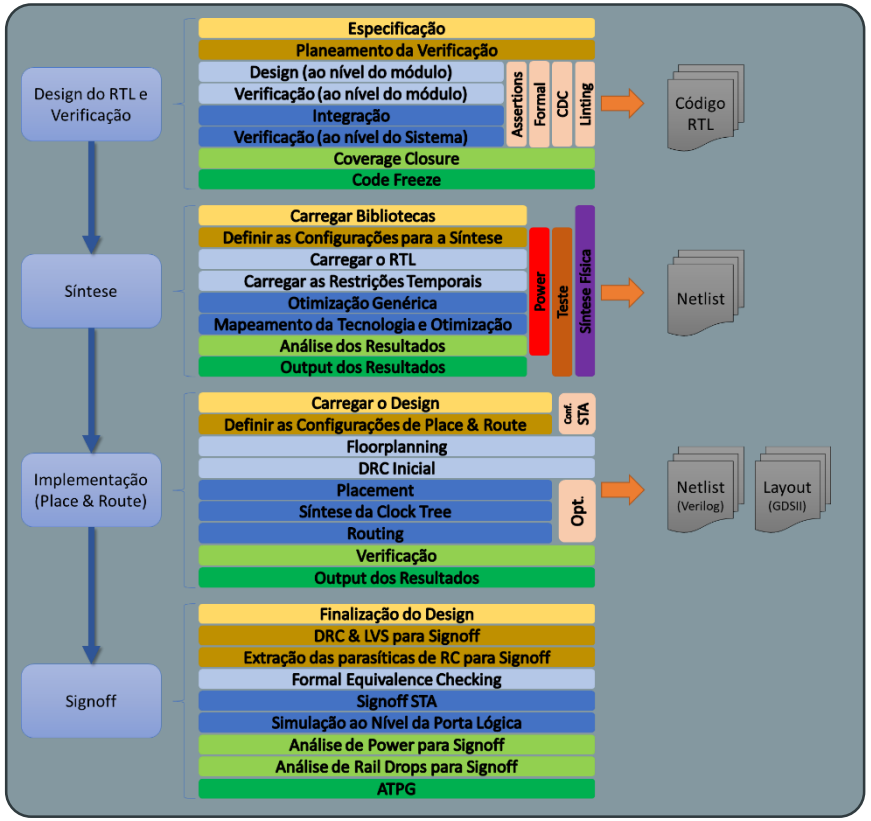
Design Flow

- ASIC
  - The Place & Route process is an iterative process which implies multiple re-runs in order to achieve the best possible design layout
  - The typical steps involved in the Place & Route phase are:
    - Floorplanning: defines the design's area, macros placement (e.g. memories or other black boxes and IP's), and padring
    - Power planning: defines the power ring and special routing
    - Placement: places the logic cells on the defined area



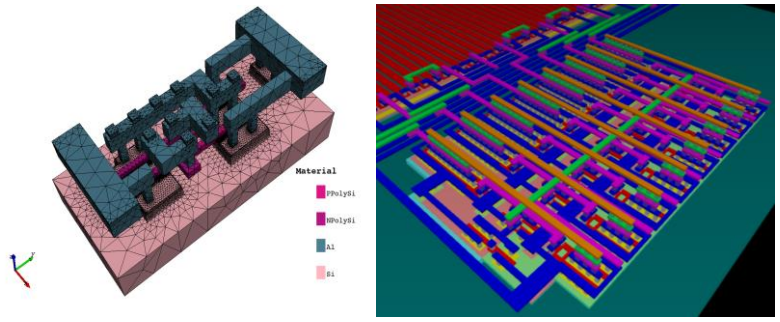
Design Flow

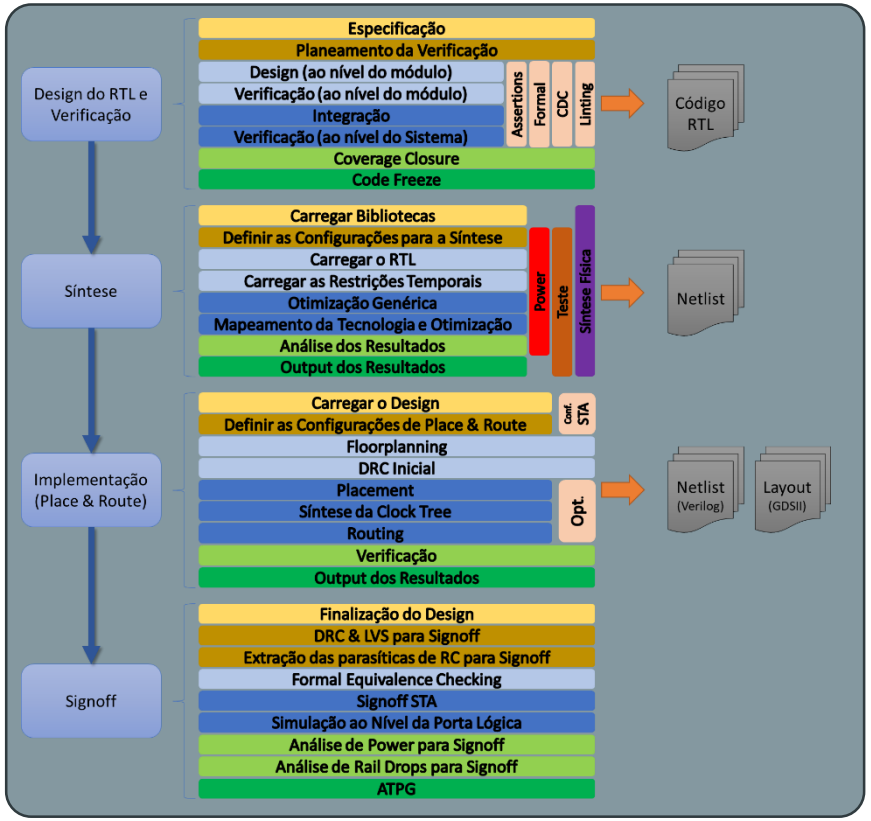
- ASIC
  - The typical steps involved in the Place & Route phase are:
    - Clock Tree Synthesis (CTS) and Clock Tree Optimization (CTO): creation of the clock tree that will feed the clock signal to the design (depending on the design, multiple clock trees can be created)
    - Routing: creates all the nets that route all the logic cells, creates metal vias and metal wires according to the specification defined by the designer (for example, clock nets might have a different specification compared to signal nets)
    - Design finishing: filler insertion and pad insertion for bounding



Design Flow

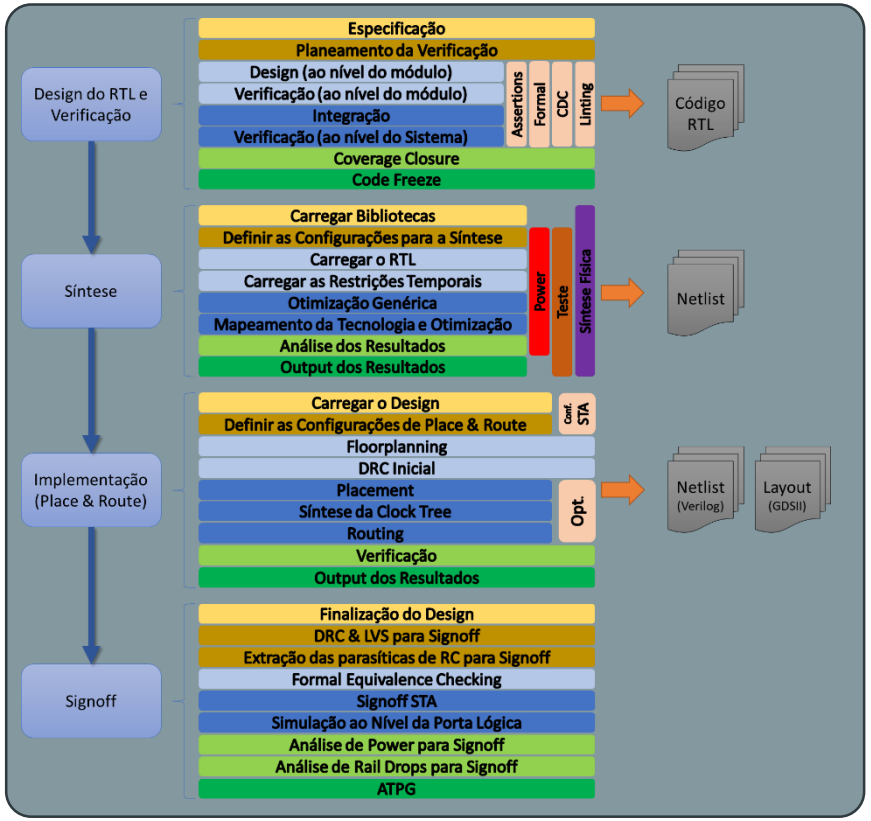
- ASIC
  - The typical steps involved in the Place & Route phase are:
    - Export the design: GDSII file generation, creation of the Verilog Netlist for the design's layout, creation of a set of files with timing and parasitic information (SDF and SPEF). These files are required for signoff tests
  - A GDSII file is a binary file representing planar geometries, text, and other information regarding the design's layout.





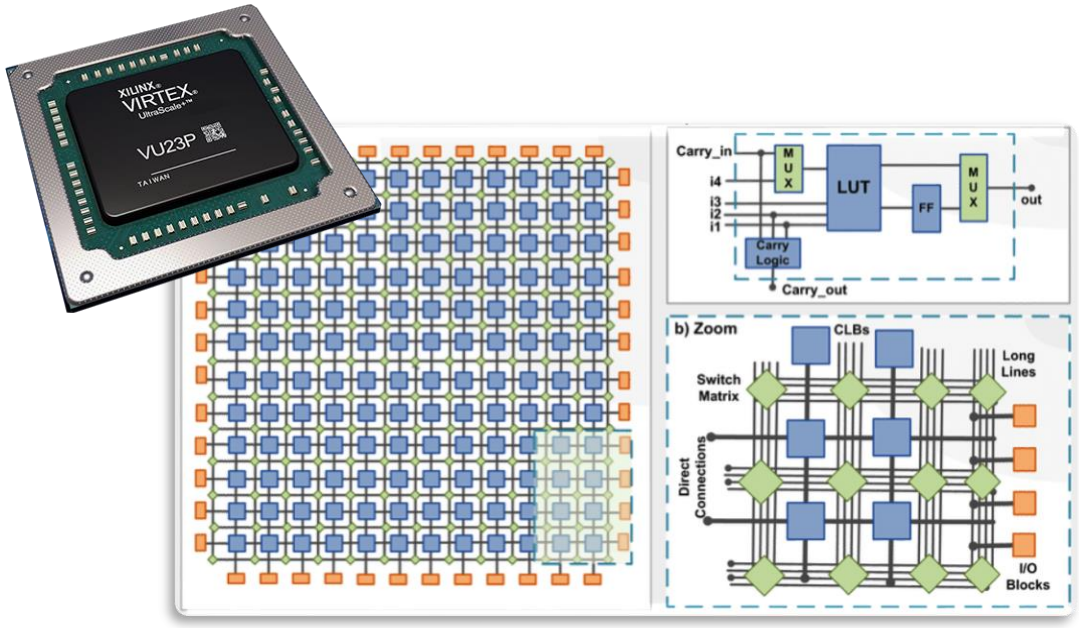
Design Flow

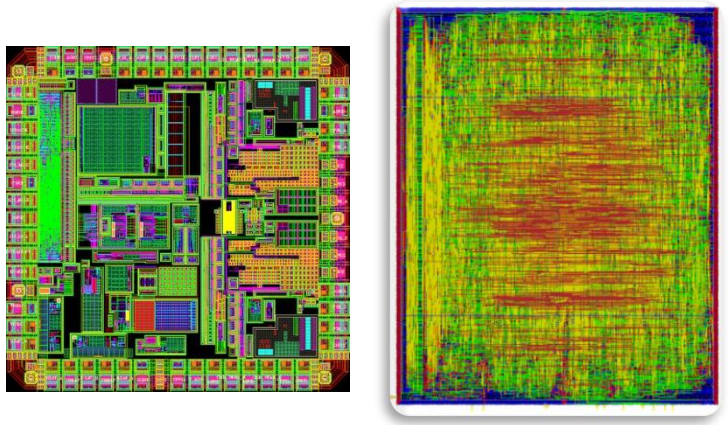
- ASIC
  - Finally the flow ends with the design's Tape Out (Signoff)
  - This has the objective to verify that the design is ready for fabrication and includes multiple tests like DRC, LVS, ERC, STA, formal equivalence checking, power analysis, simulations with delay information

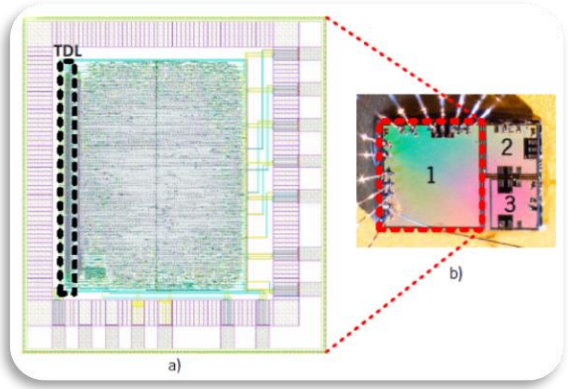


Design Flow

FPGA Vs ASIC

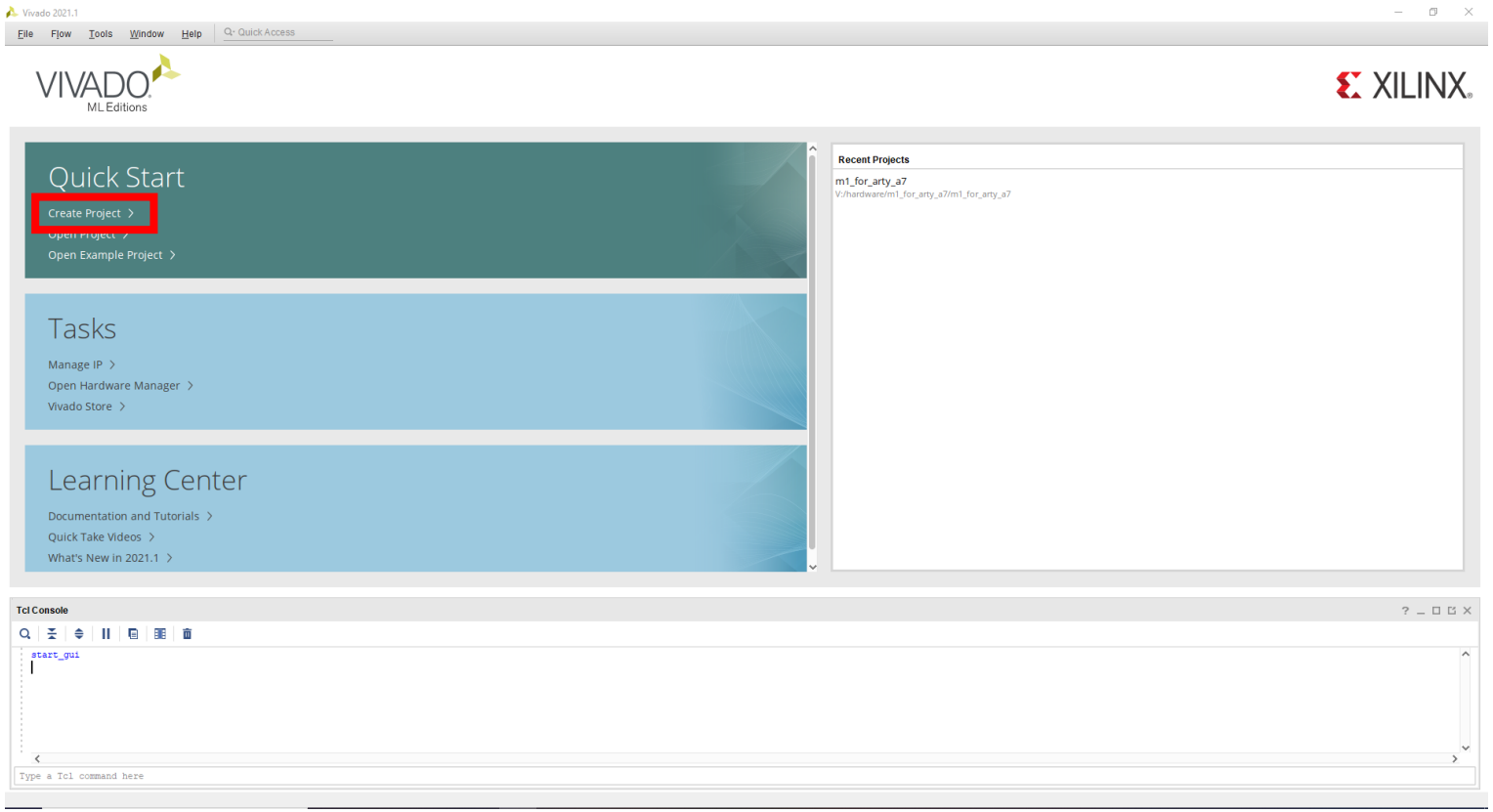






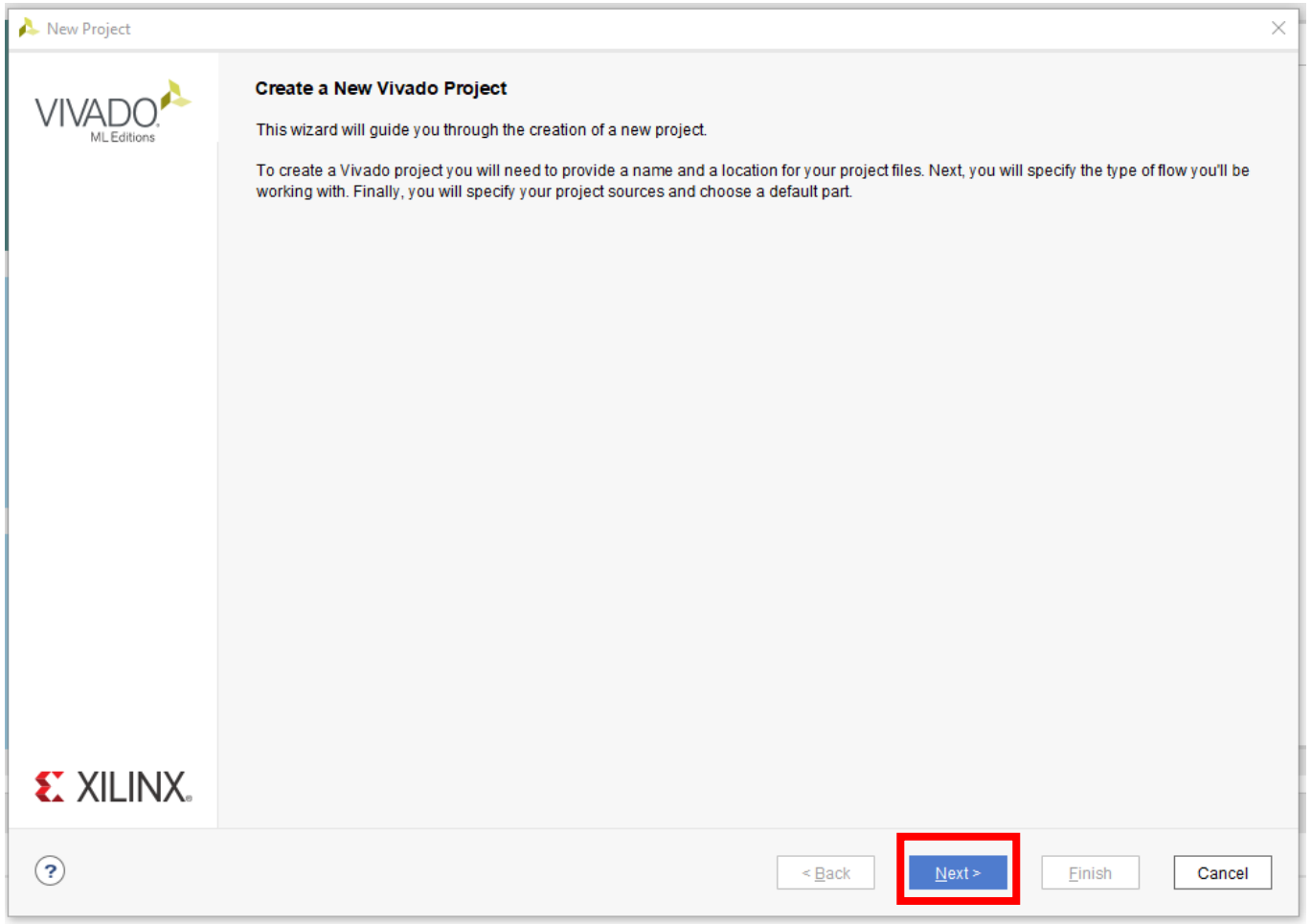
Design Flow

- Creating a project
- Open Vivado and click “Create Project”



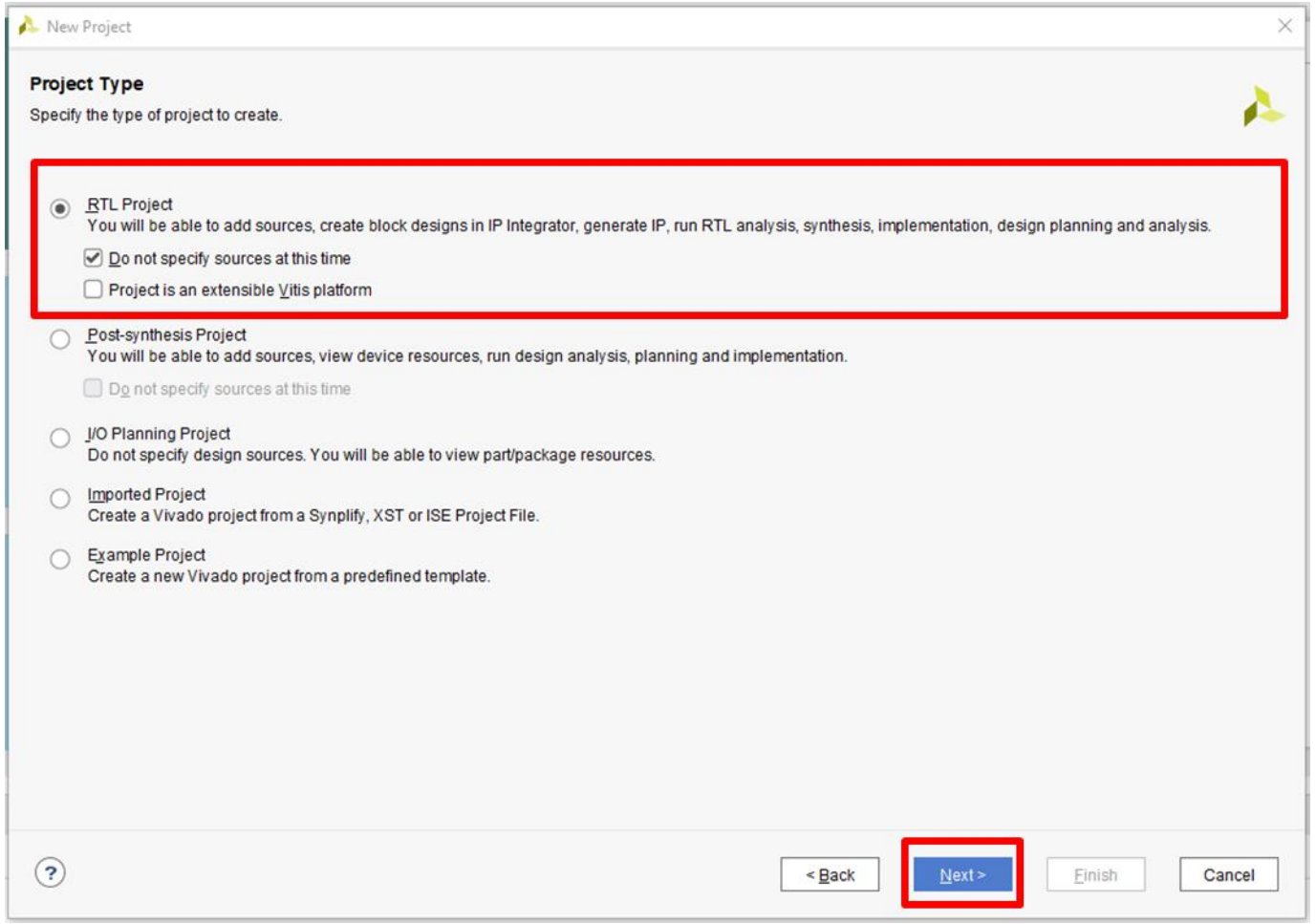
Design Flow

- This will start the New Project Wizard
- Next



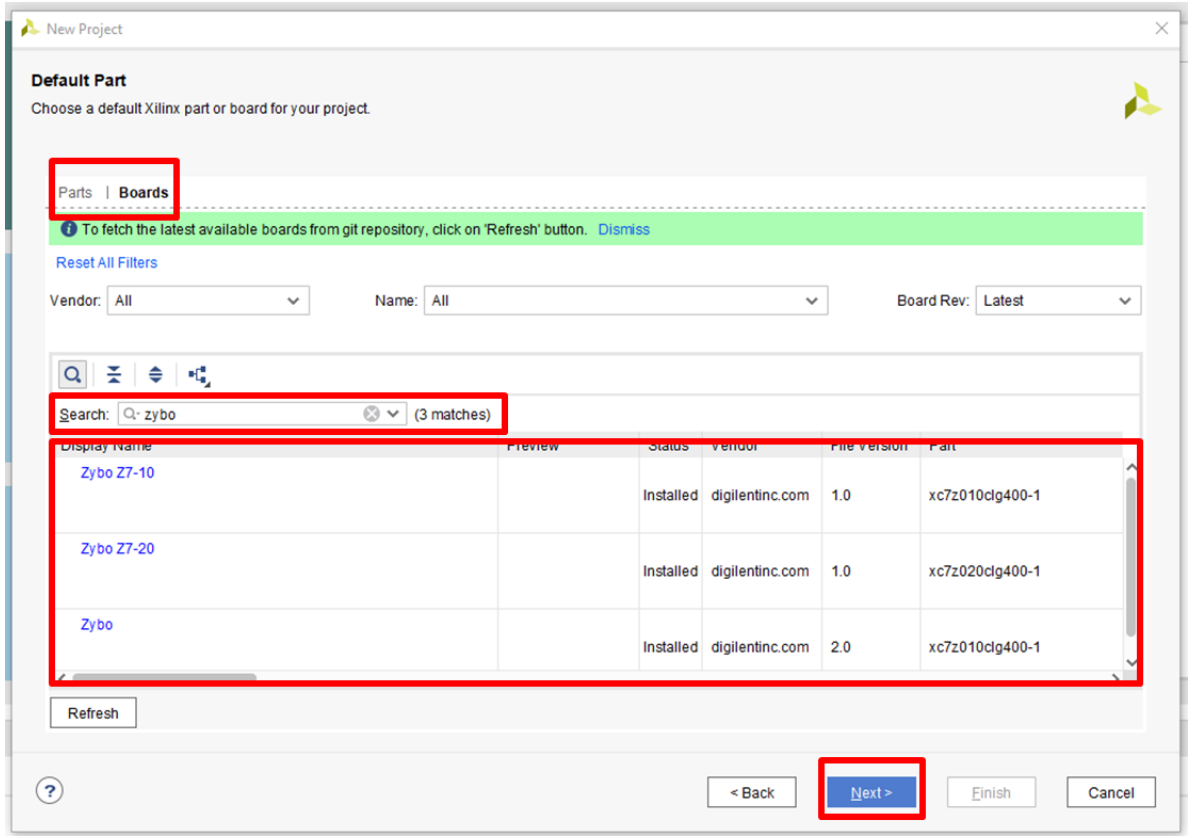
Design Flow

- Select RTL Project and do not specify sources at this time
- Next



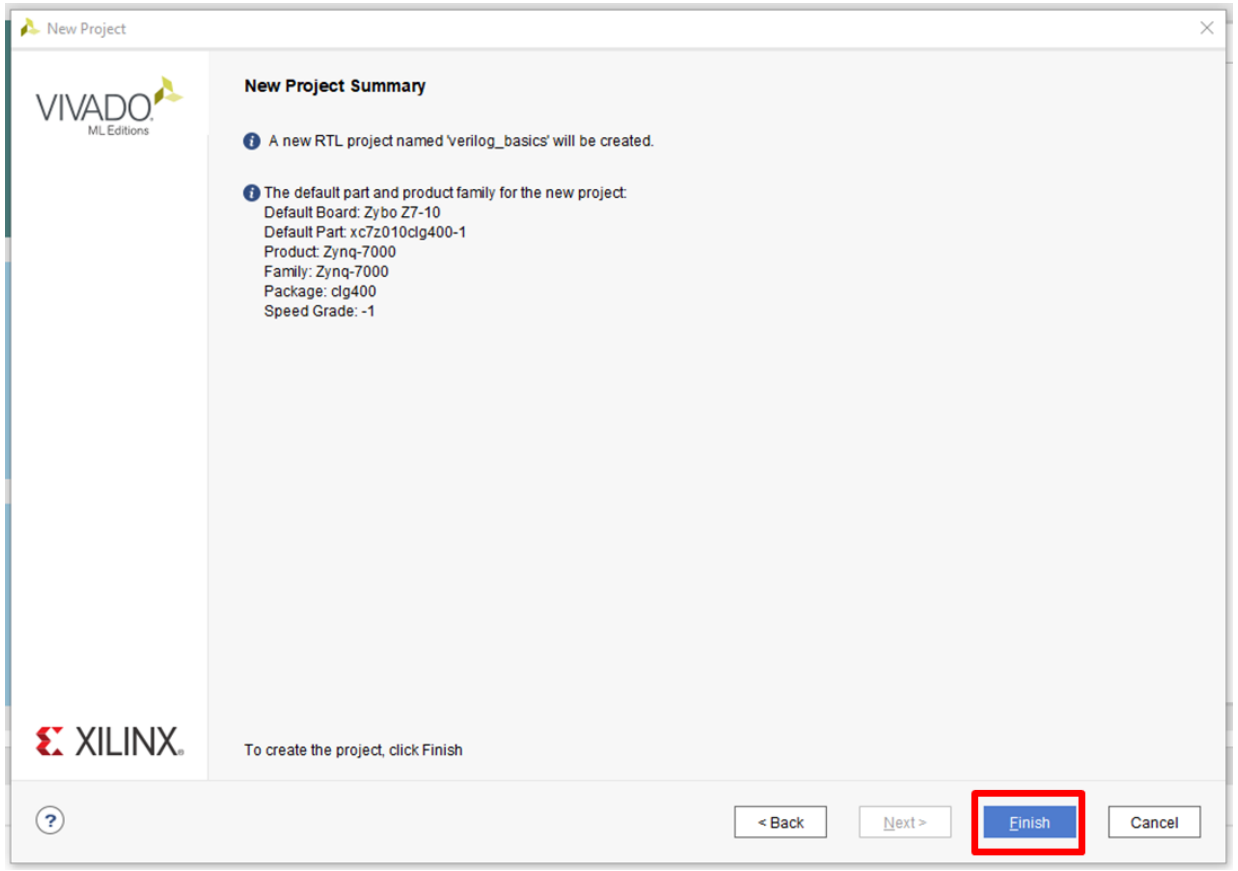
Design Flow

- On the Default Part window select Boards
- Type “zybo” on the search bar
- The boards you just installed from Digilent should appear
- Select the correct board you are using
- Next



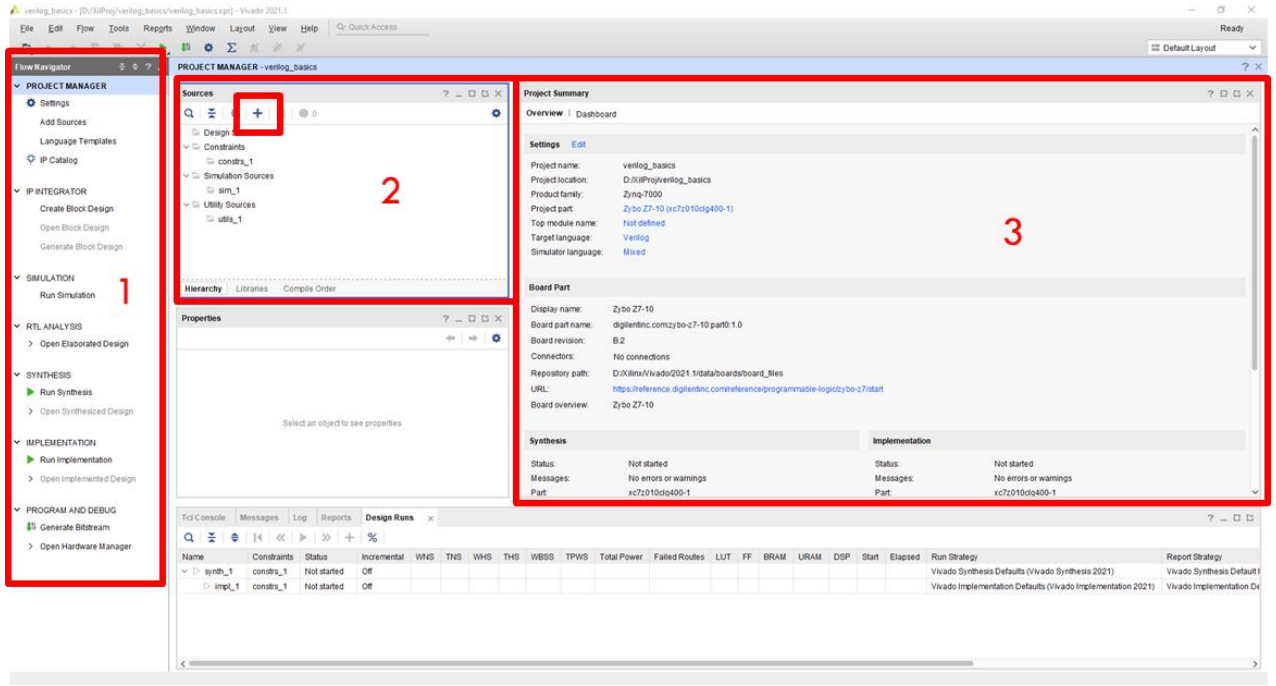
Design Flow

- All set and ready to go.
- Check the information window and click “Finish”



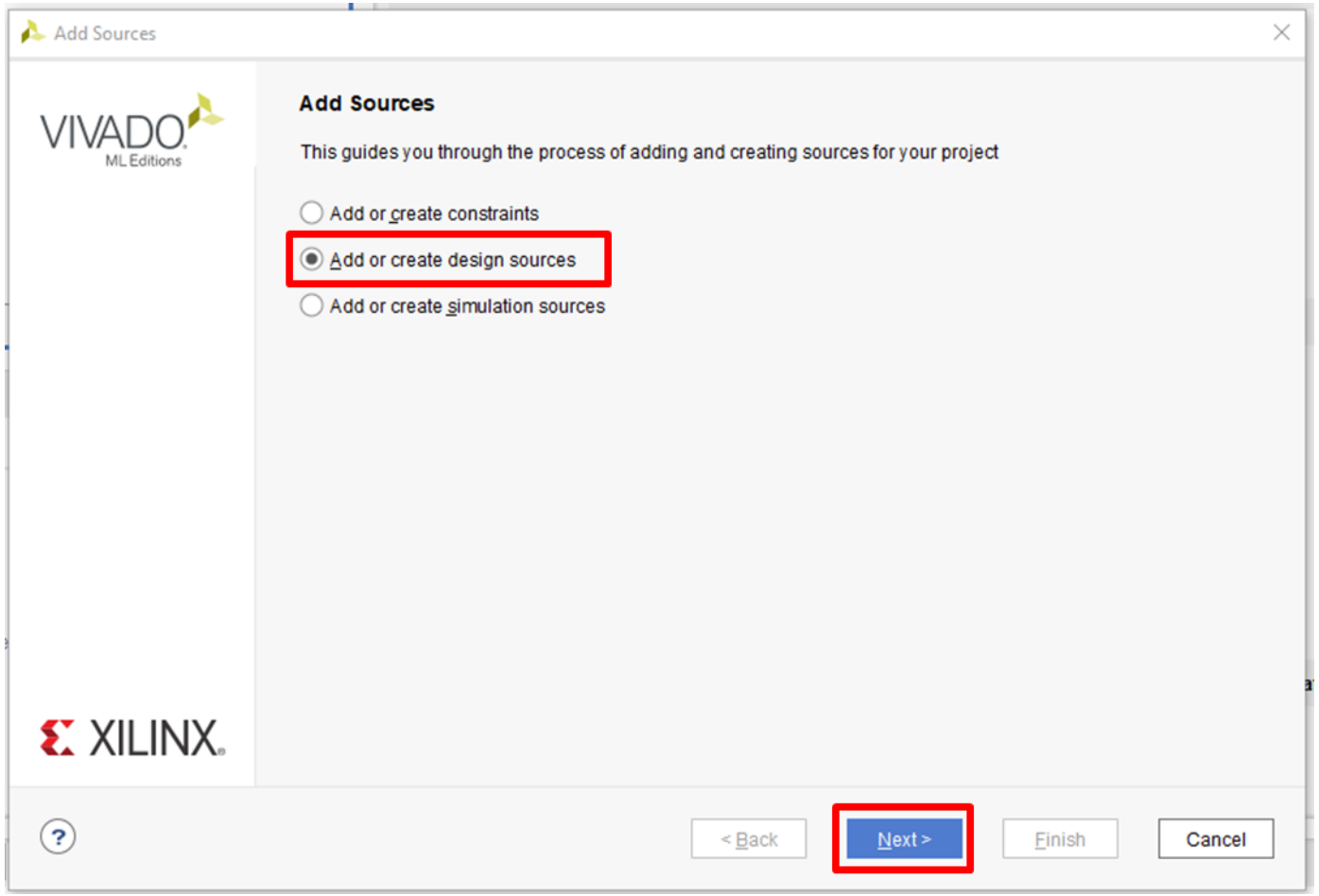
Design Flow

- You should see Vivado Project Summary window with information about your design (3)
- You can also see the source files of your design (2) and the design flow menus (1)
- Add a new file by clicking on the “+” icon



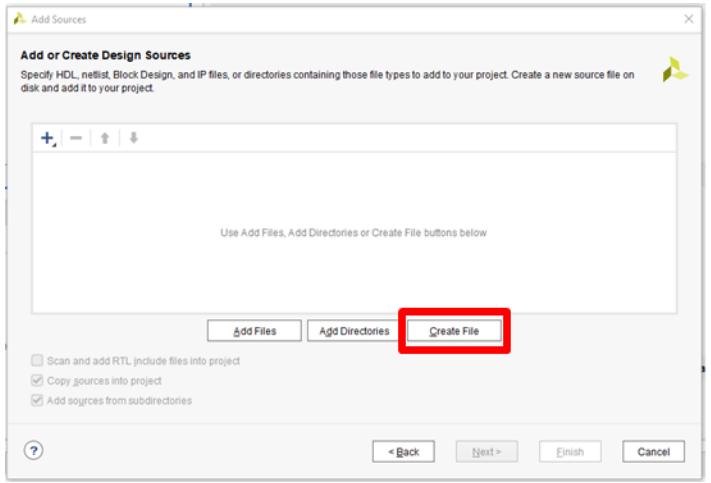
Design Flow

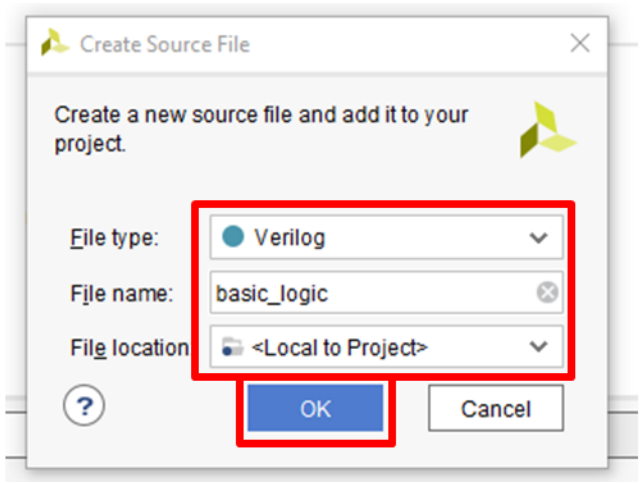
- Select the “add or create design sources”
- Next

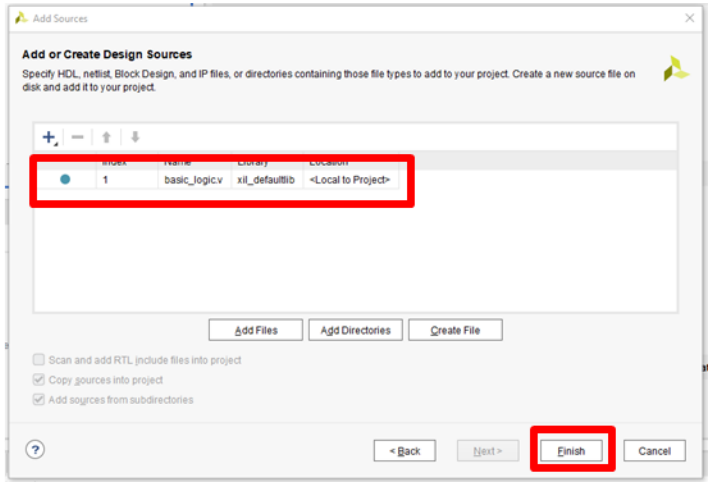


Design Flow

- Select “Create File”, then select “Verilog” for the file type, the file name, leave the directory as it is and click ok. You should see your file now. Click “Finish”

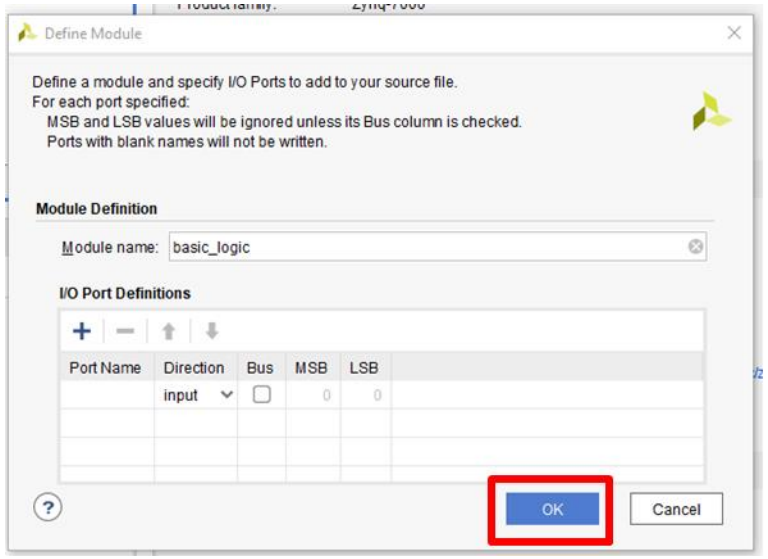


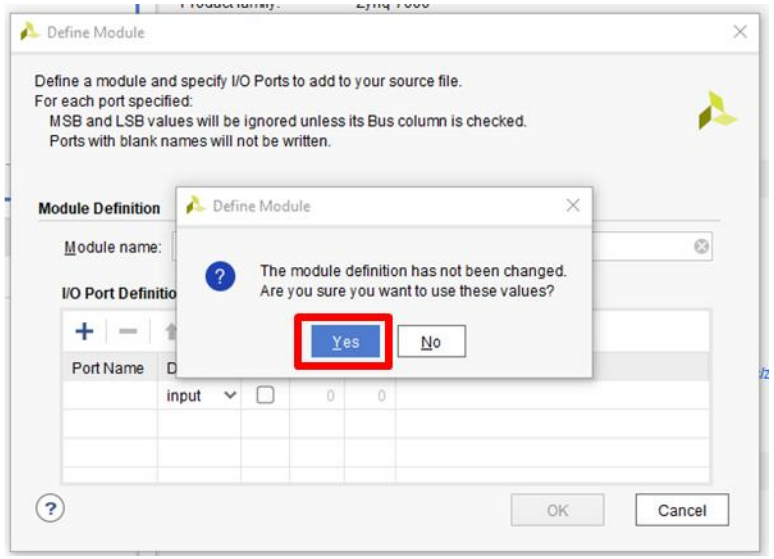




Design Flow

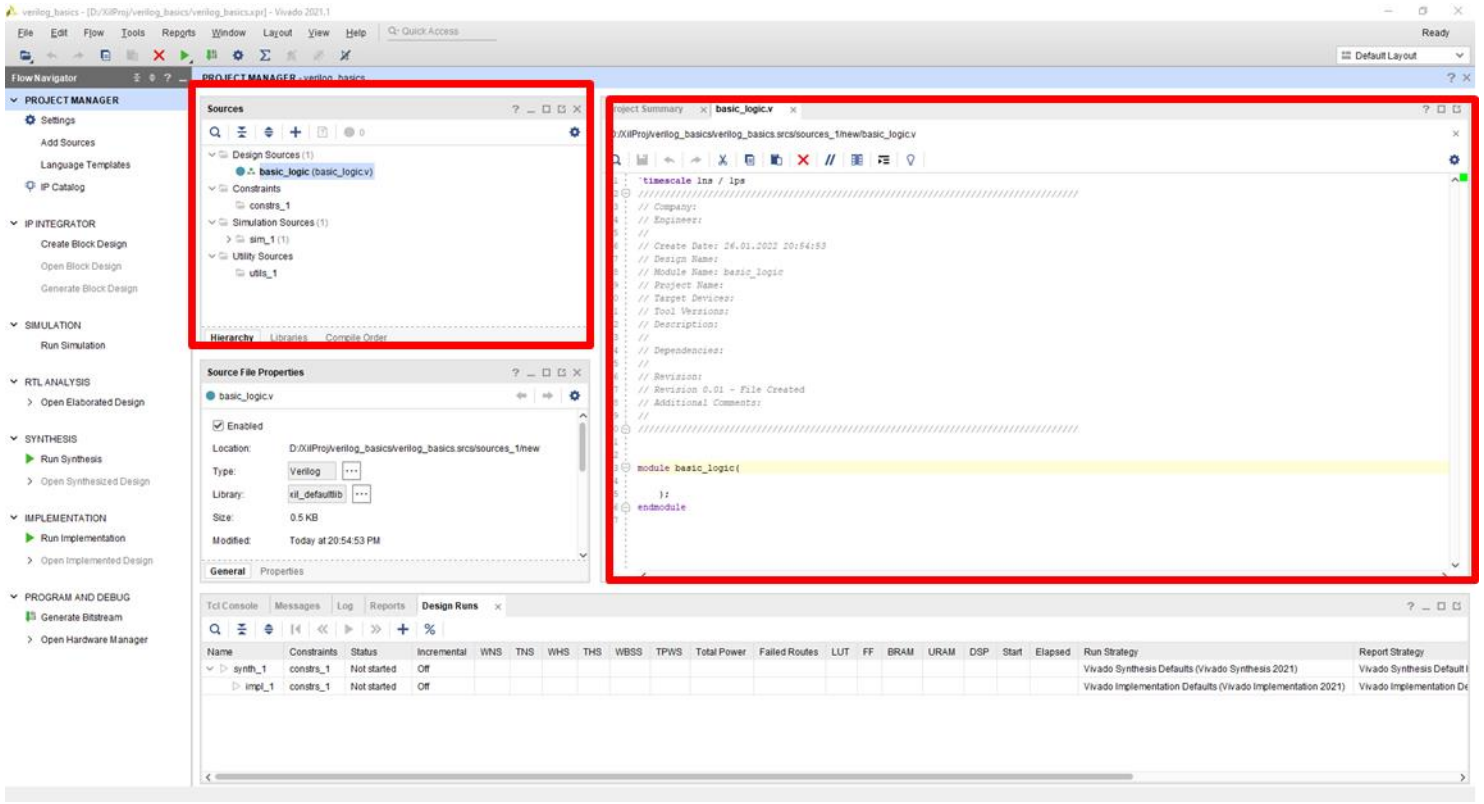
- On the next menu, leave everything as it is and click “Ok”
- You could have created your design ports here but for now we just want an empty module.
- Acknowledge your decision by clicking “Yes”





Design Flow

- We now have a new module on our source window
- Double click on the file to open it



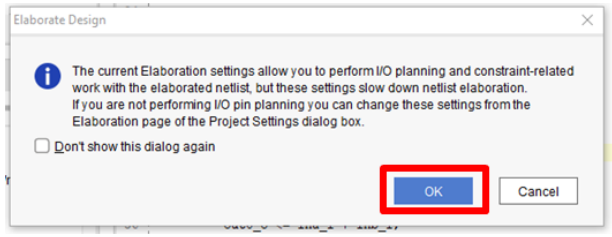
Design Flow

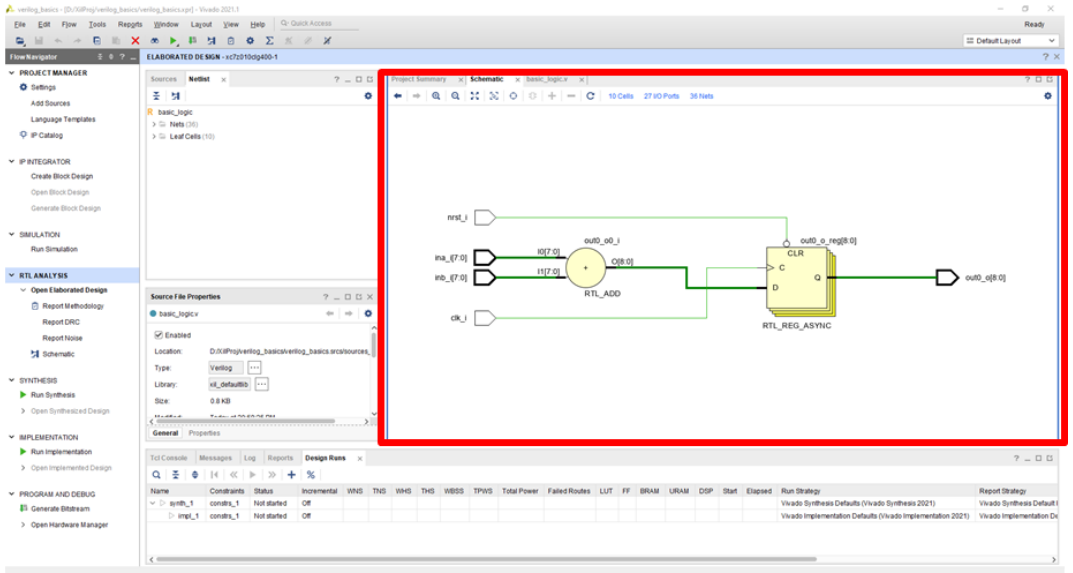
- Let's make a simple 8-bit adder
- Write the following code on the module we have just created

```
23  module basic_logic(clk_i, nrst_i, ina_i, inb_i, out0_o);
24
25  input clk_i;
26  input nrst_i;
27  input [7:0] ina_i;
28  input [7:0] inb_i;
29  output reg [8:0] out0_o;
30
31  always @(posedge clk_i or negedge nrst_i) begin
32      if(~nrst_i) begin
33          out0_o <= {9{1'b0}};
34      end
35      else begin
36          out0_o <= ina_i + inb_i;
37      end
38  end
39
40
41  endmodule
42
```

Design Flow

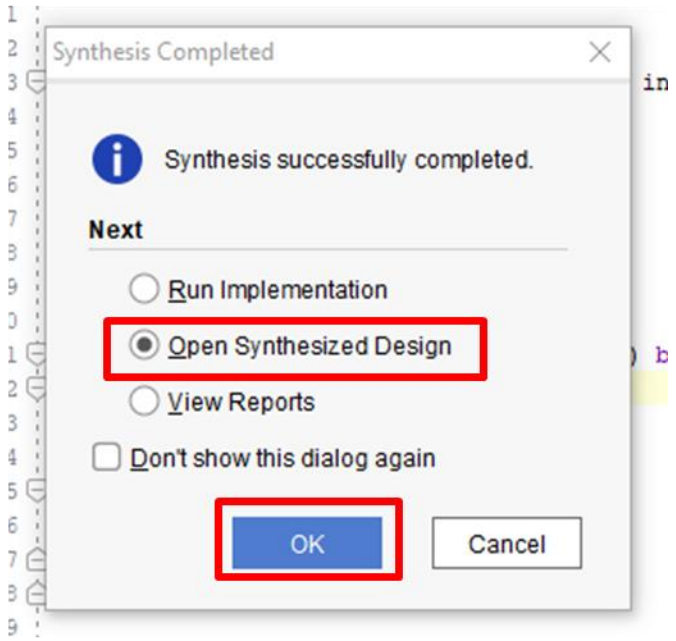
- Time to check our logic. Select “Open Elaborated Design”
- Select Ok on the next window
- You should see a schematic similar to the one presented on the picture on the below. Take your time to understand it





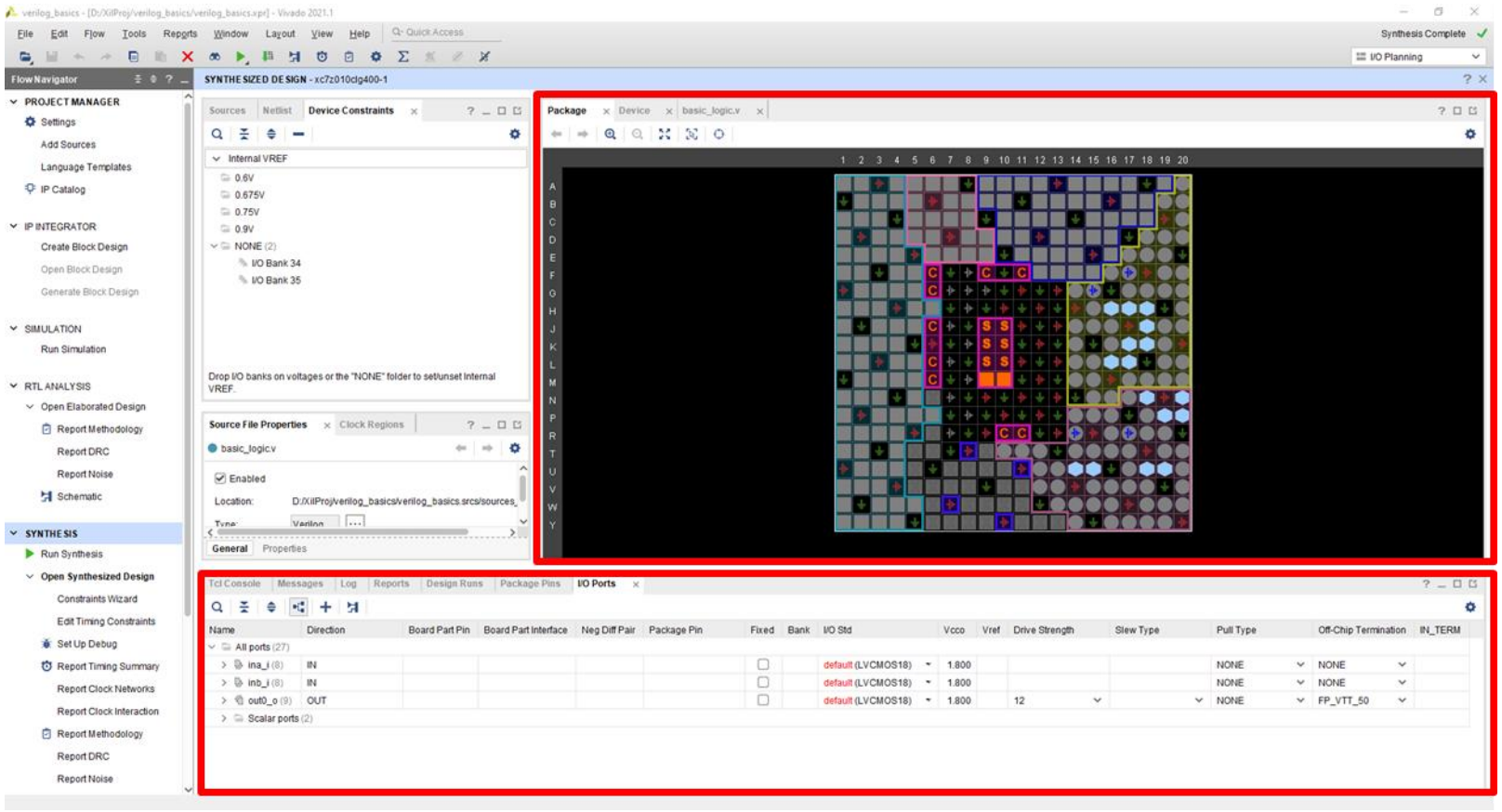
Design Flow

- Next step is to synthesize your design. Click on “Run Synthesis” and then “Ok”. When it finishes, select “Open Synthesized Design” and click “OK”



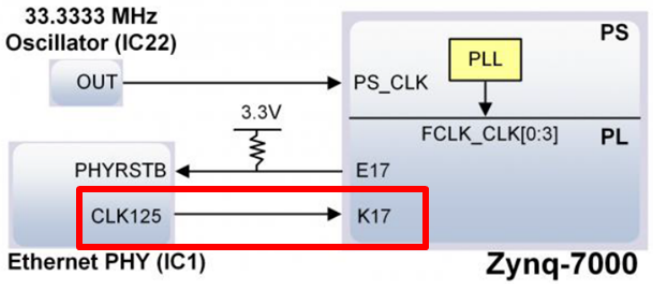
Design Flow

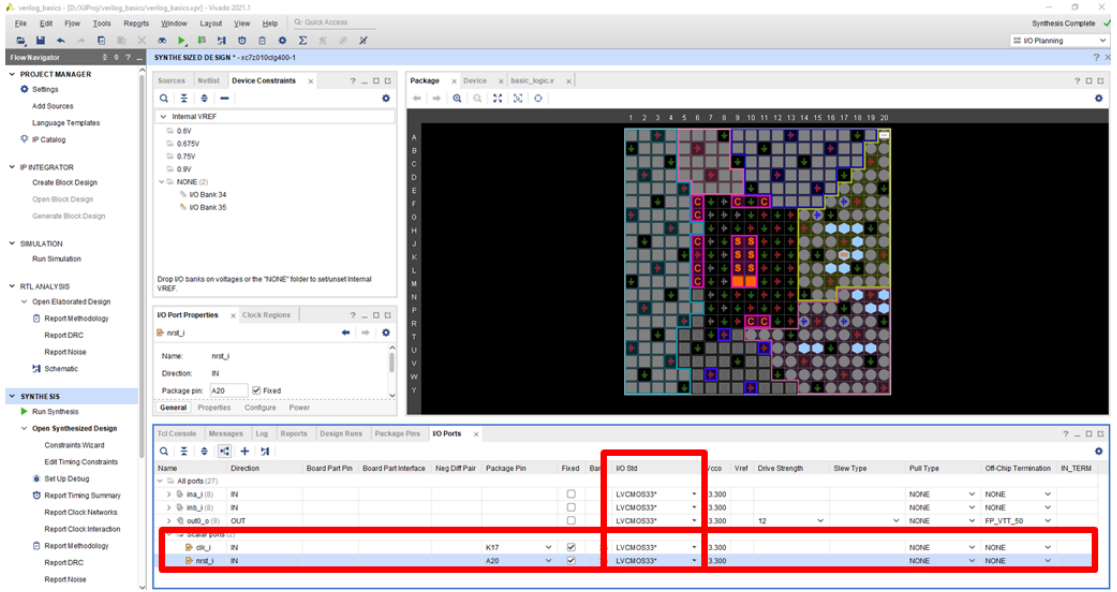
- You should see the “Package” view on the right of the “Synthesized Design” window and the “I/O Ports” menu at the bottom



Design Flow

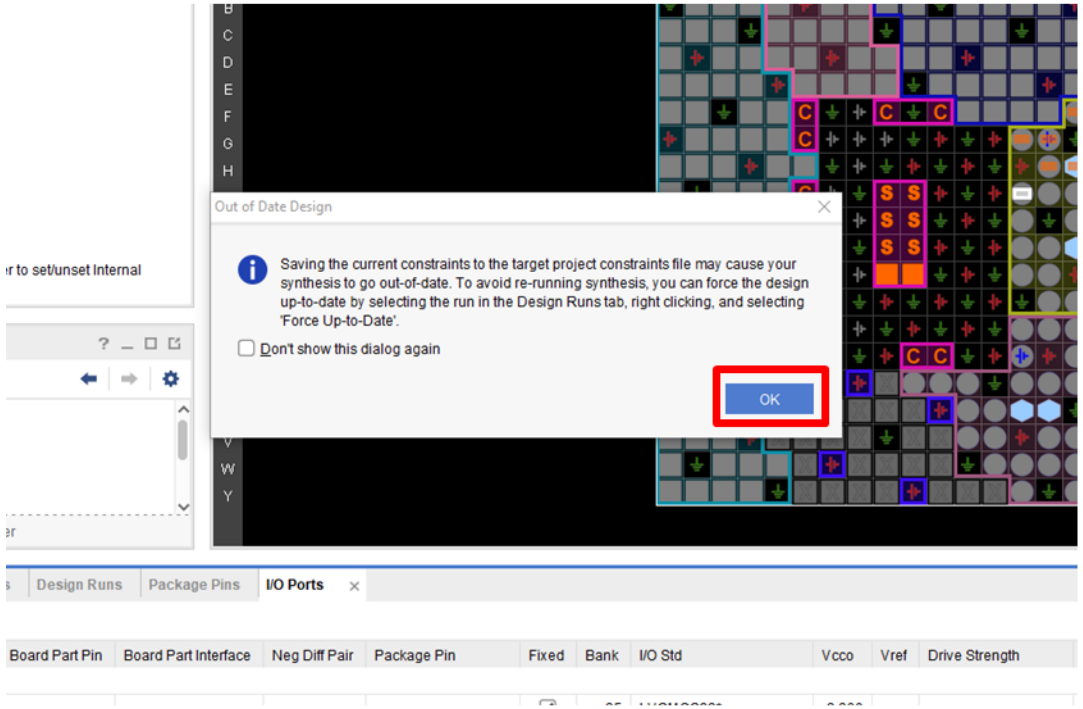
- Fill in the ports with FPGA Pins of your choice. Only the clock must be tied to a specific pin which you can find on the Zybo documentation. For the Zybo board, the PL clock pin is the K17. This clock is a 125MHz clock generated by the Ethernet PHY integrated circuit (IC)
- Make sure “I/O Std” is set to LVCMOS33 for all I/Os

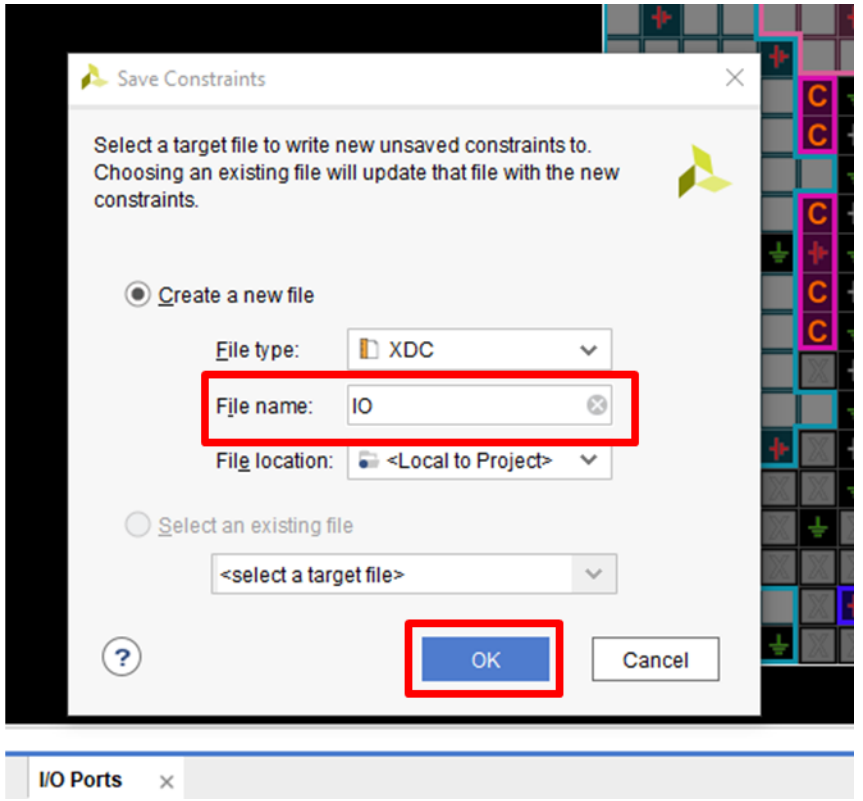




Design Flow

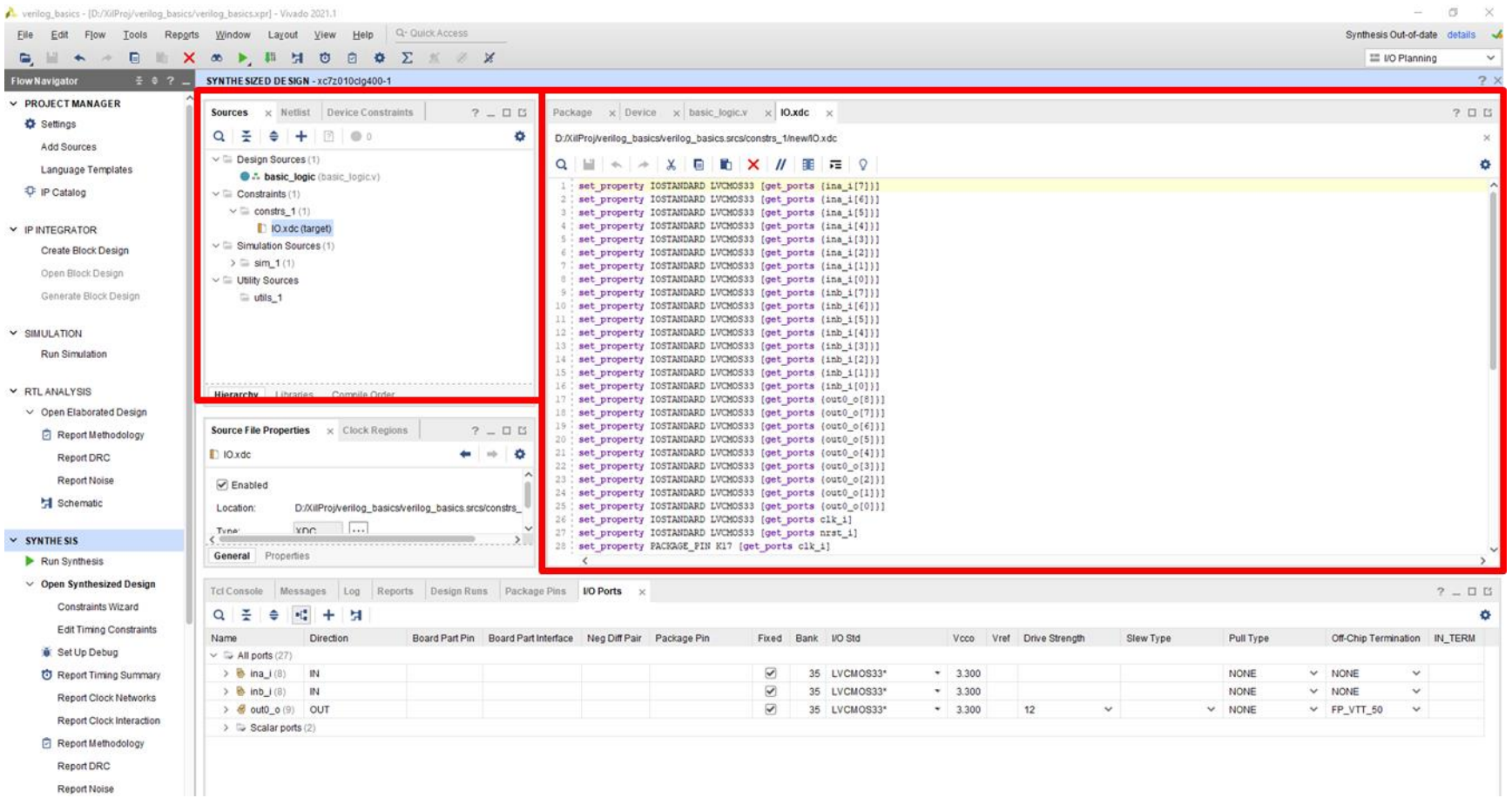
- After finishing all the I/O planning, Ctrl+s to save and then ok.
- Select a name for the constraint file you are about to create and then “OK”





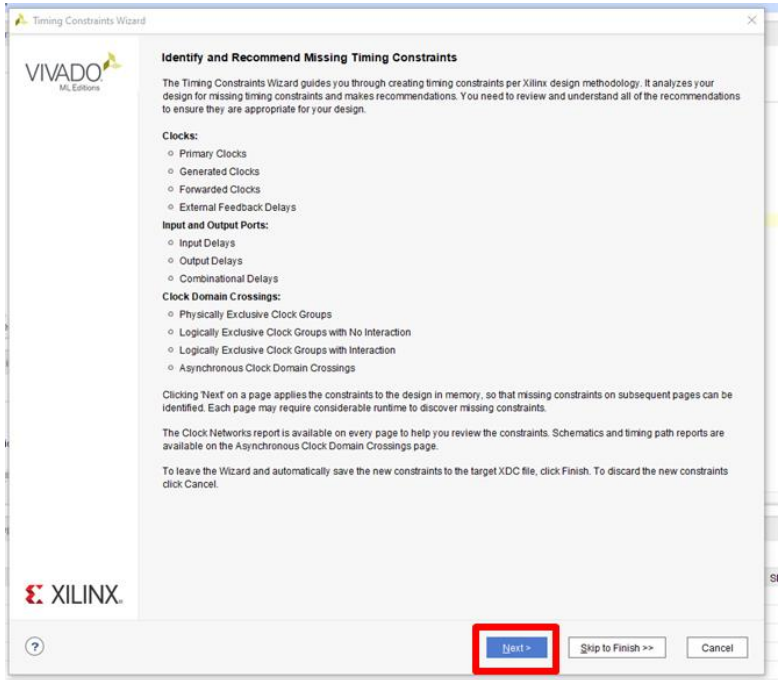
Design Flow

- Open the constraint file you just created and analyze its content



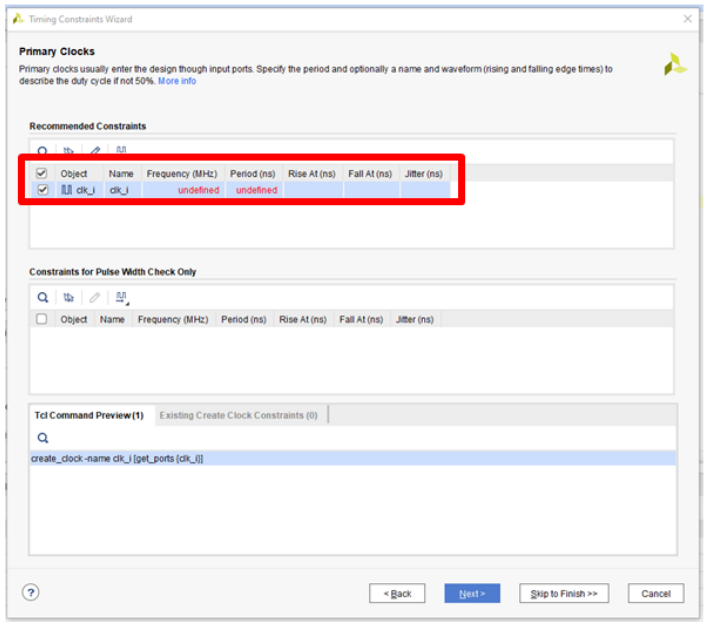
Design Flow

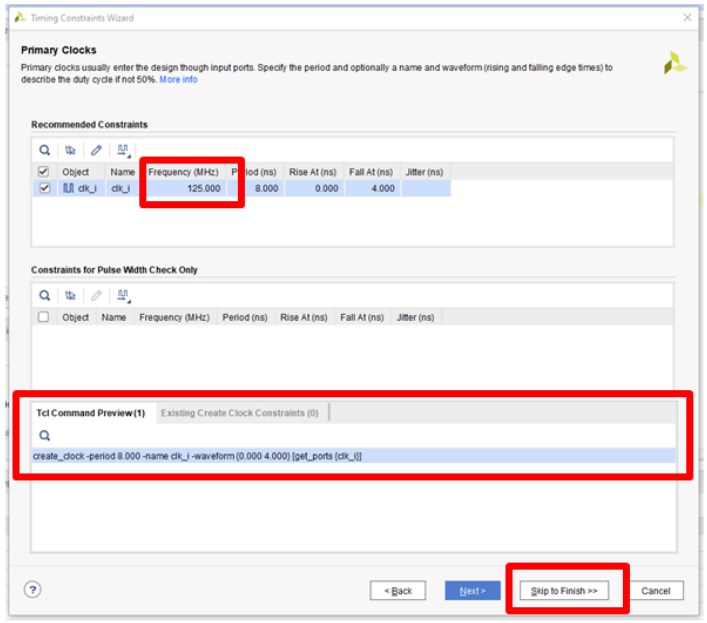
- With the IO constraints set, time to create the timing constraints.
- For this simple example we will only create a clock constraint (the most important constraint of all). All designs must have at least one clock constraint (unless pure combinatorial)
- On the Synthesis menu, select “Constraints Wizard”
- Do “Next” on the first Window

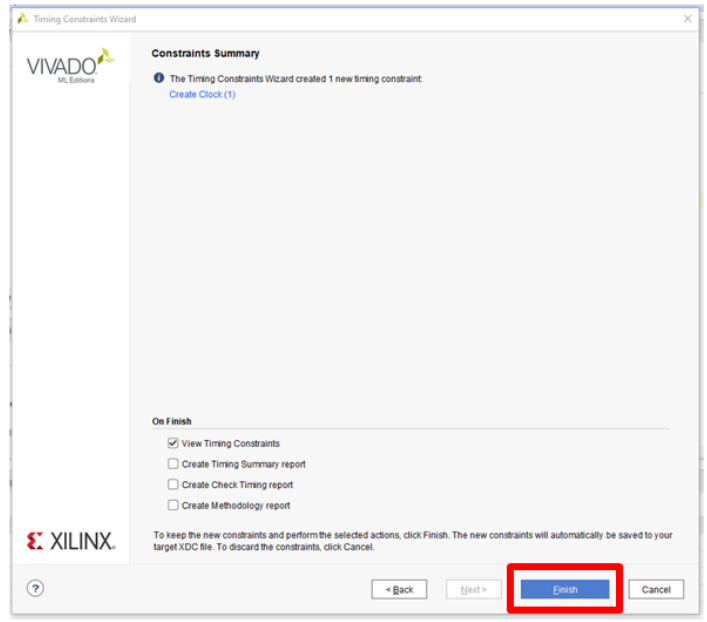


Design Flow

- Set the clk_i to match the 125MHz given by the Ethernet PHY IC
- Notice the constraint automatically created at the bottom of the window
- For now this is the only constraint that we will define. Select “Skip to Finish” and then “Finish”

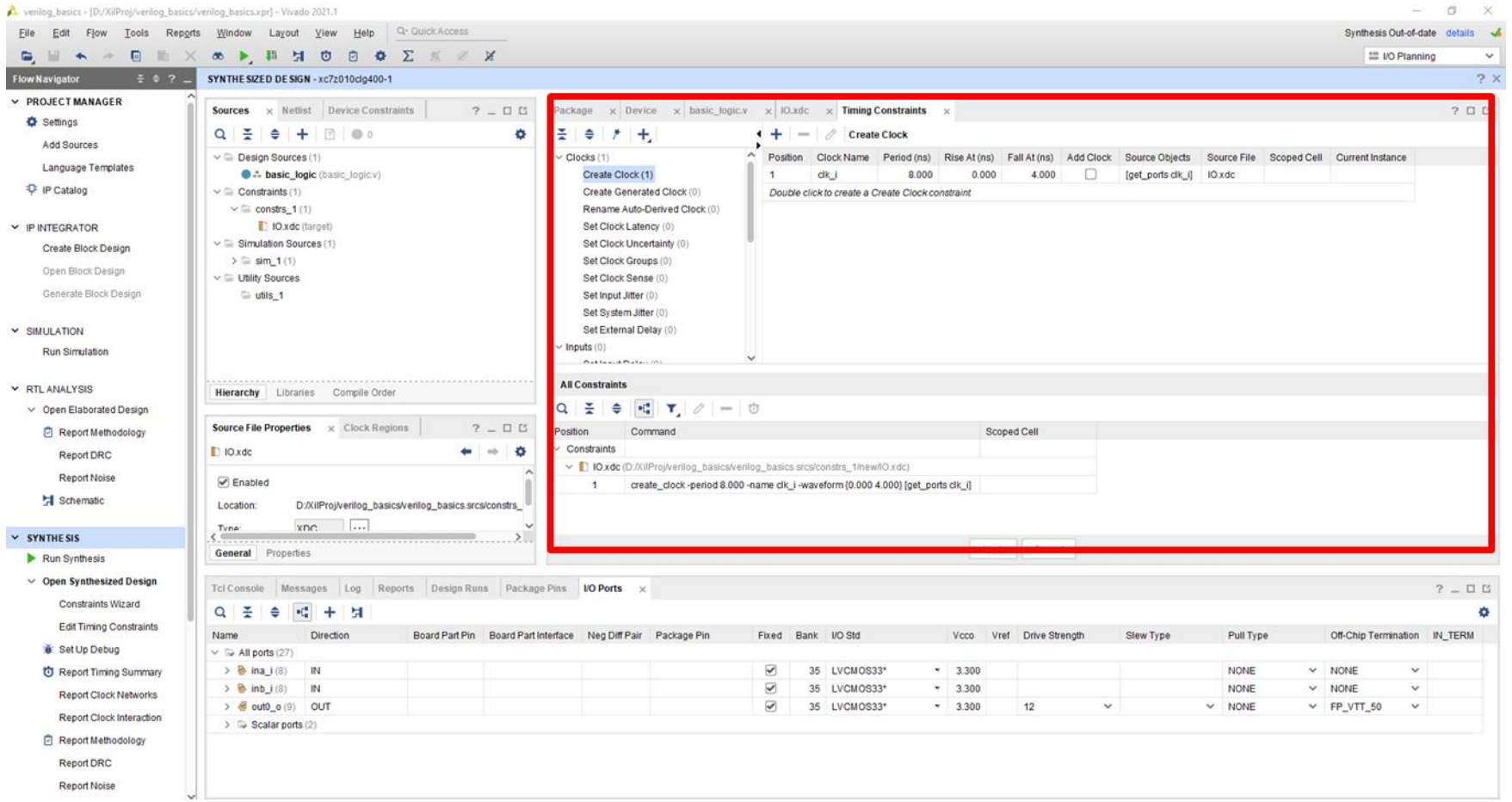






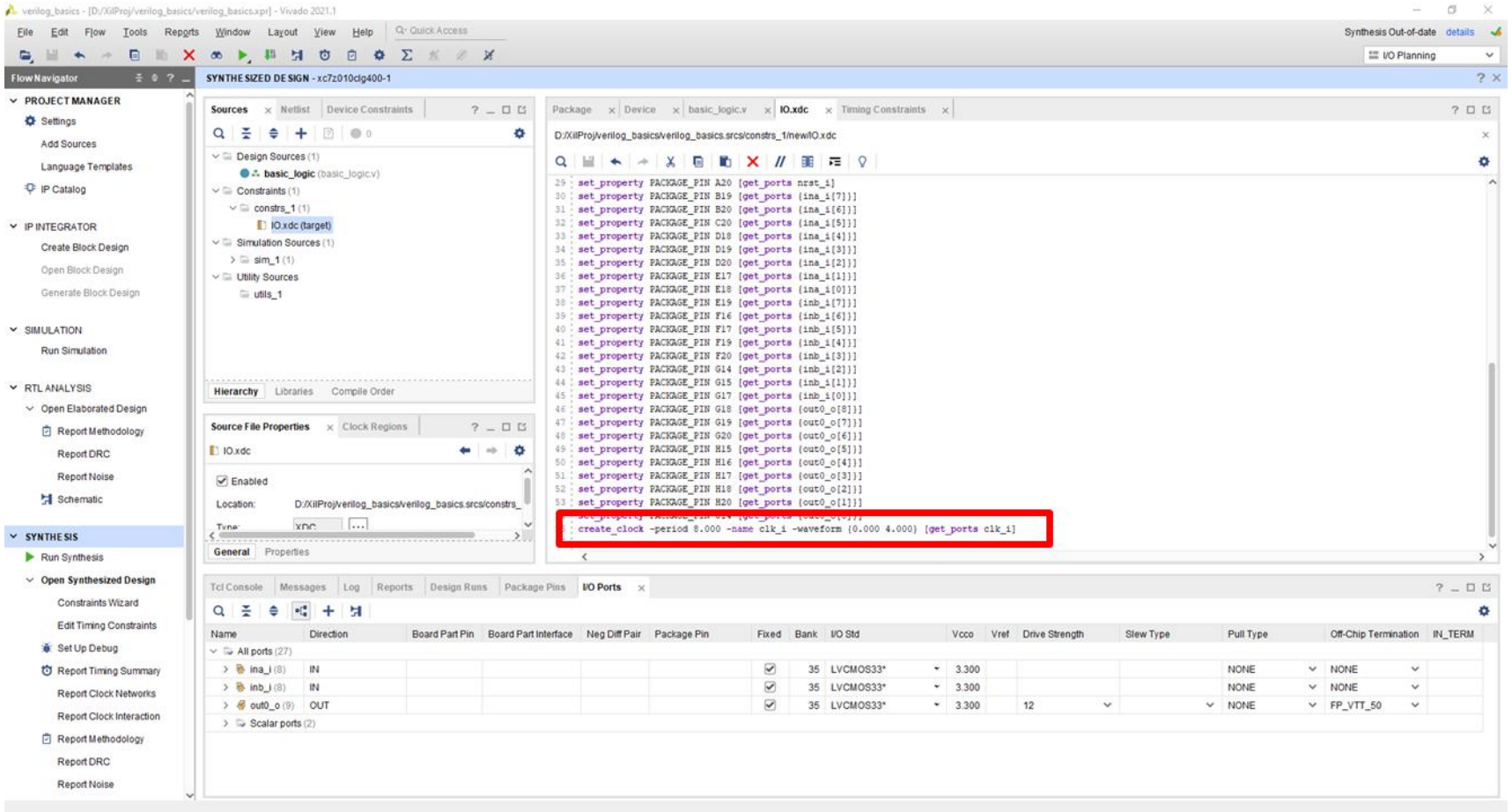
Design Flow

- You should be able to see all the design's timing constraints



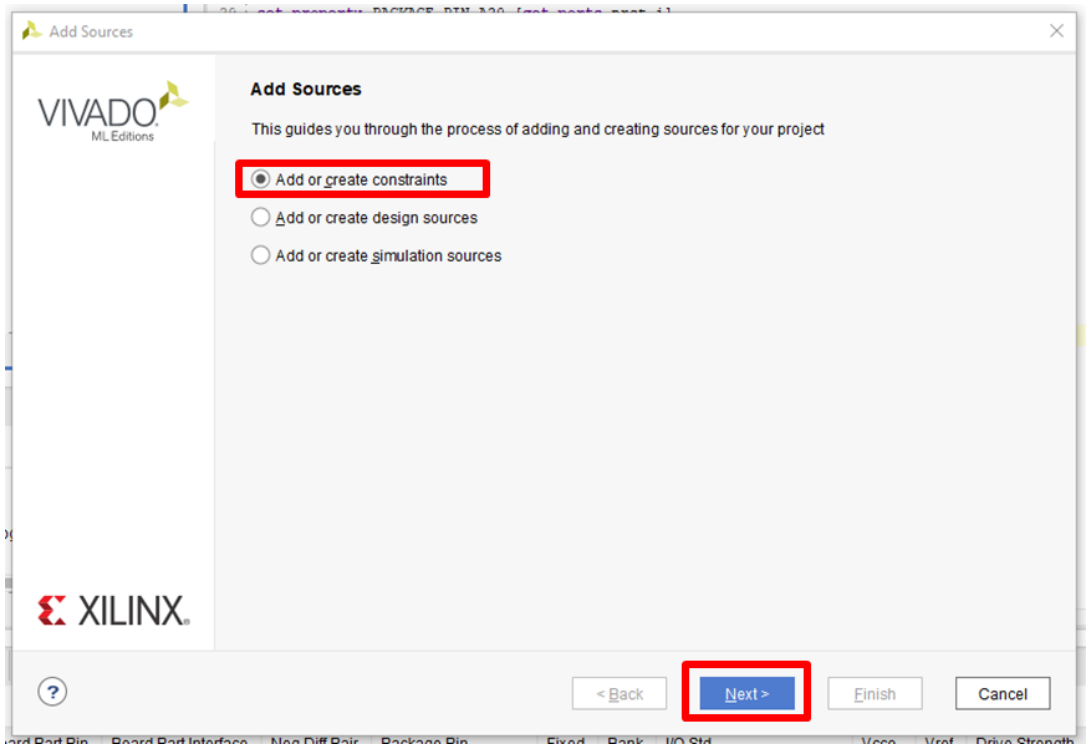
Design Flow

- You can also check your constrain file and notice a new line at the bottom of the file



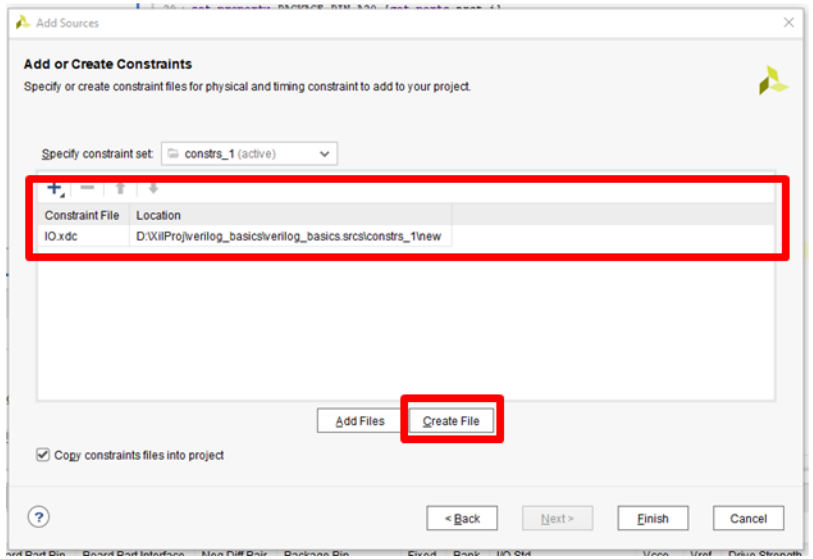
Design Flow

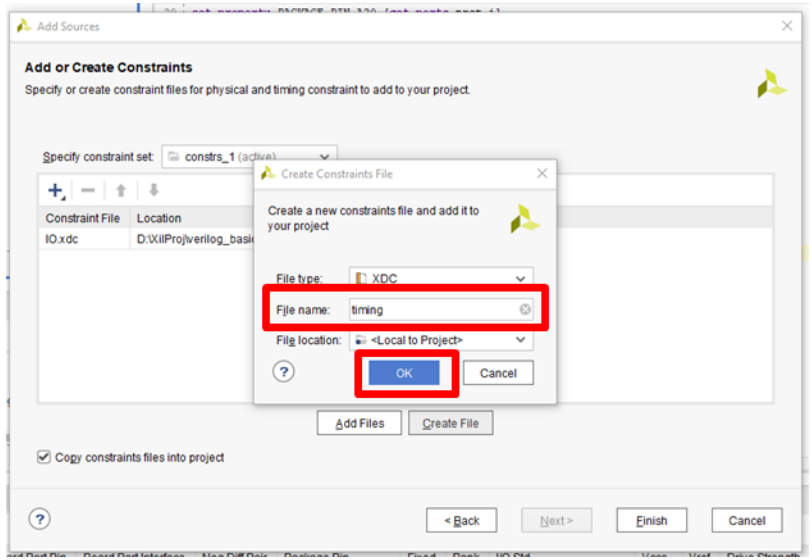
- It is a good practice to keep device physical constraints (like I/O mapping) separated from the design's timing constraints. Why?
- Lets create a new file for the timing constraints. On the source window click on the "+" icon
- Select "Add or create constraints"
- Next

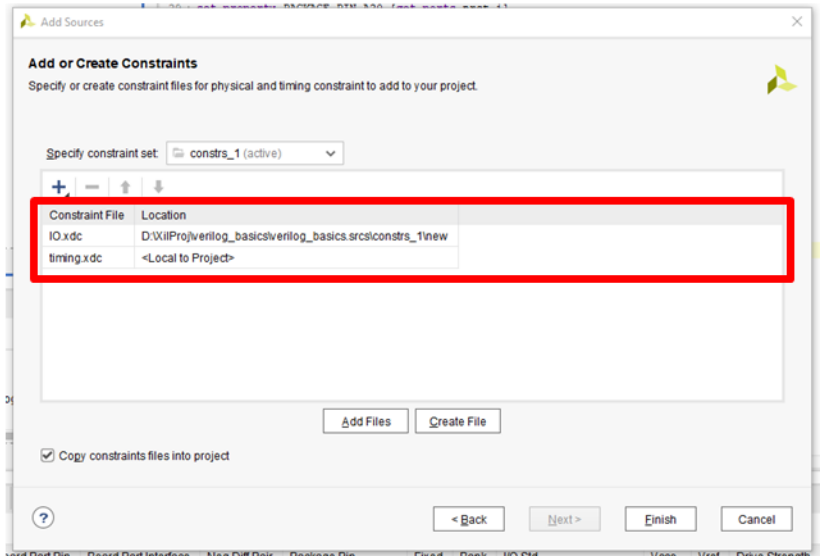


Design Flow

- You should see your IO constraint file.
- Click on “Create File” and then select a name for it. Close the windows by doing “OK” then “Finish”

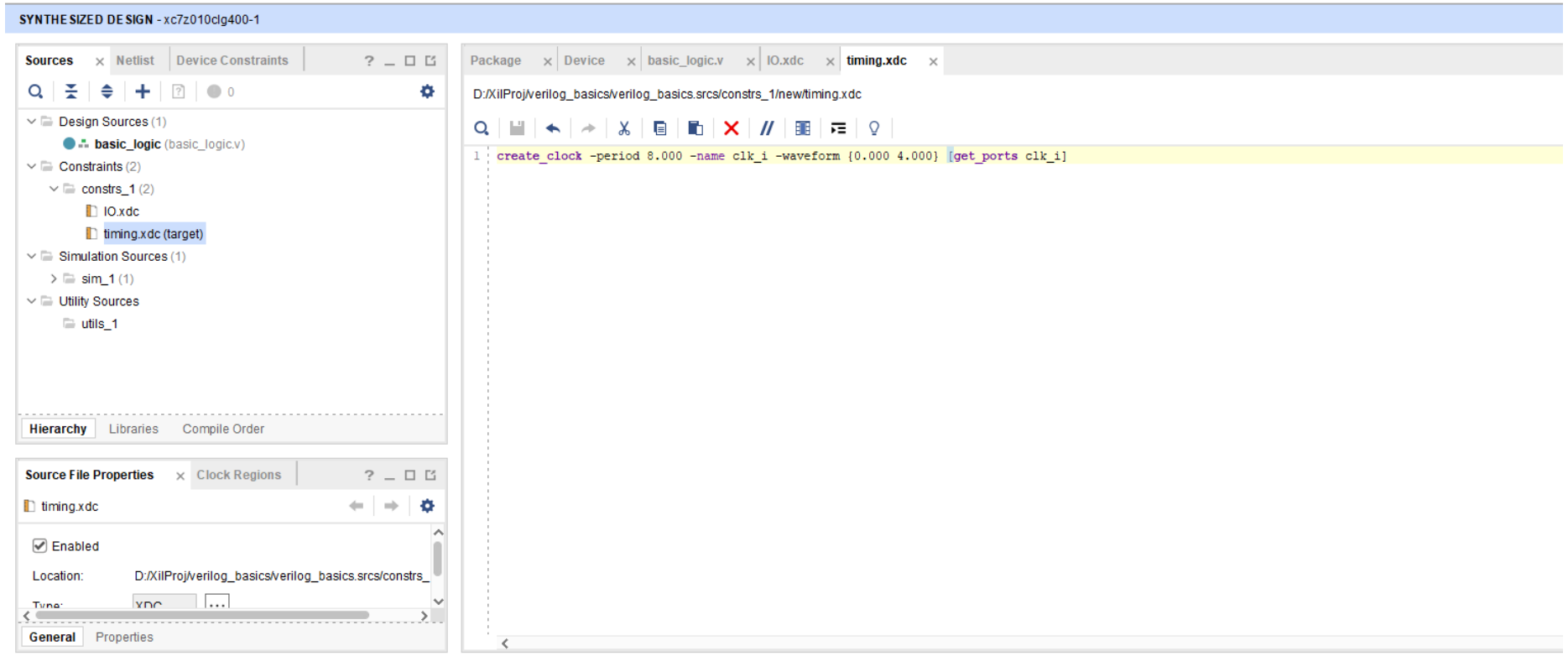






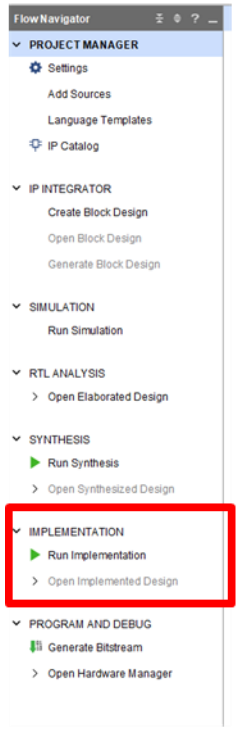
Design Flow

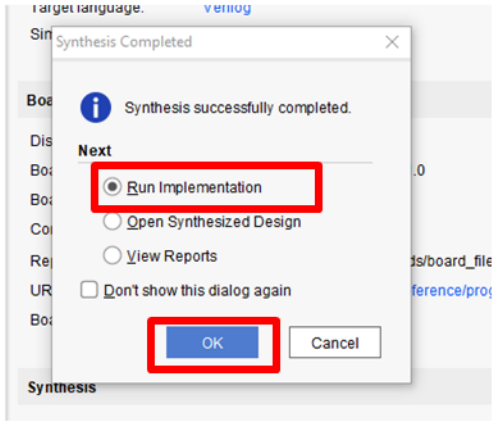
- Cut the last line on the IO constraint file and past it on the new constraint file you have just created. Right click on the file and select “Set as Target Constraint File” to make it the main constraint file

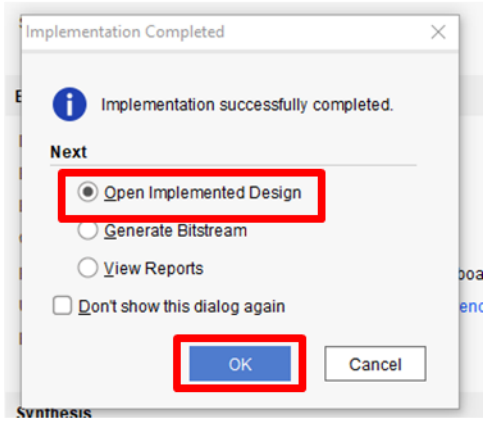


Design Flow

- Time to run the implementation phase. Click on “Run implementation”. If the design needs to be re-synthesized, when synthesis completes select “Run Implementation” and “OK”. Select “Open Implemented Design” when the process is completed

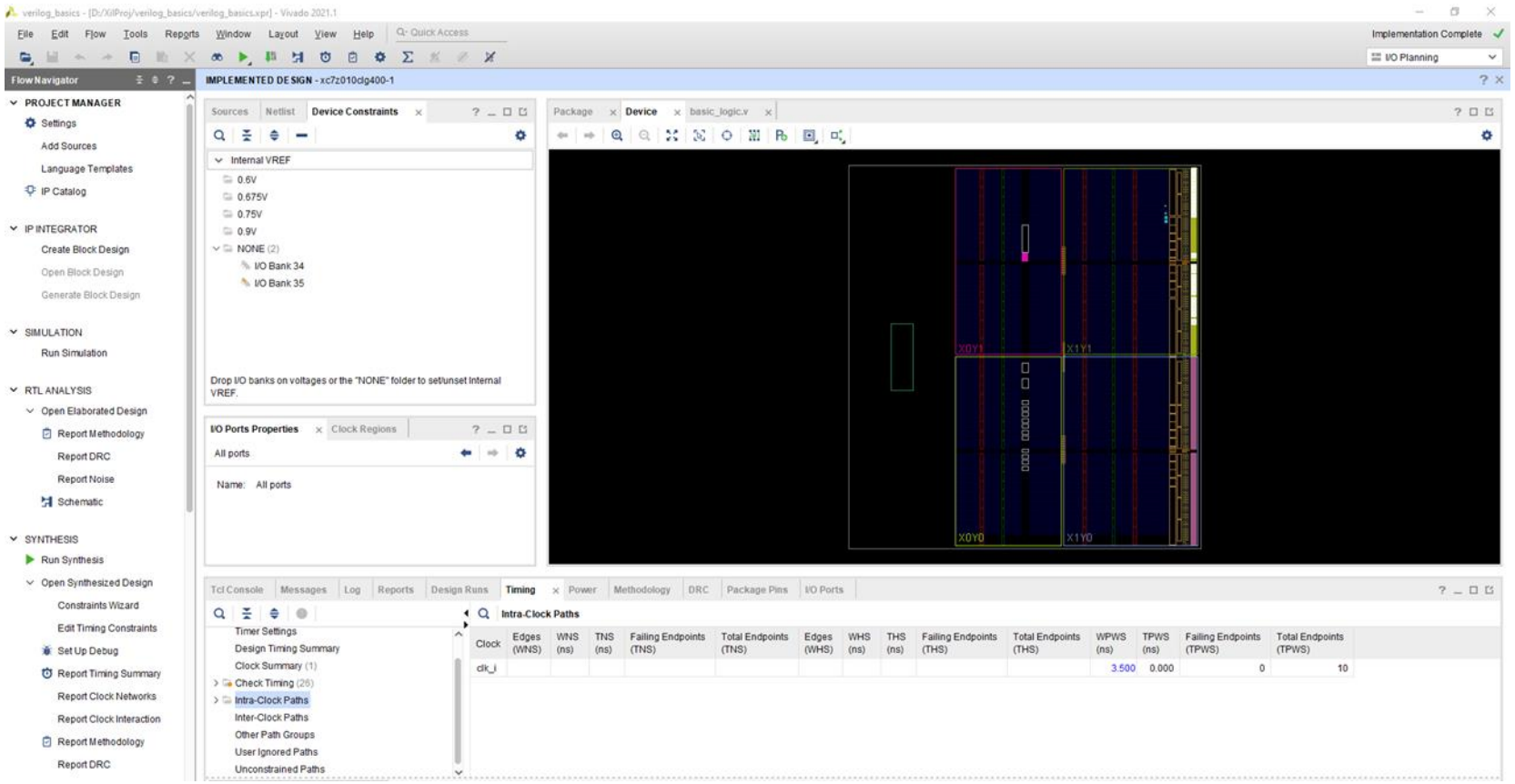






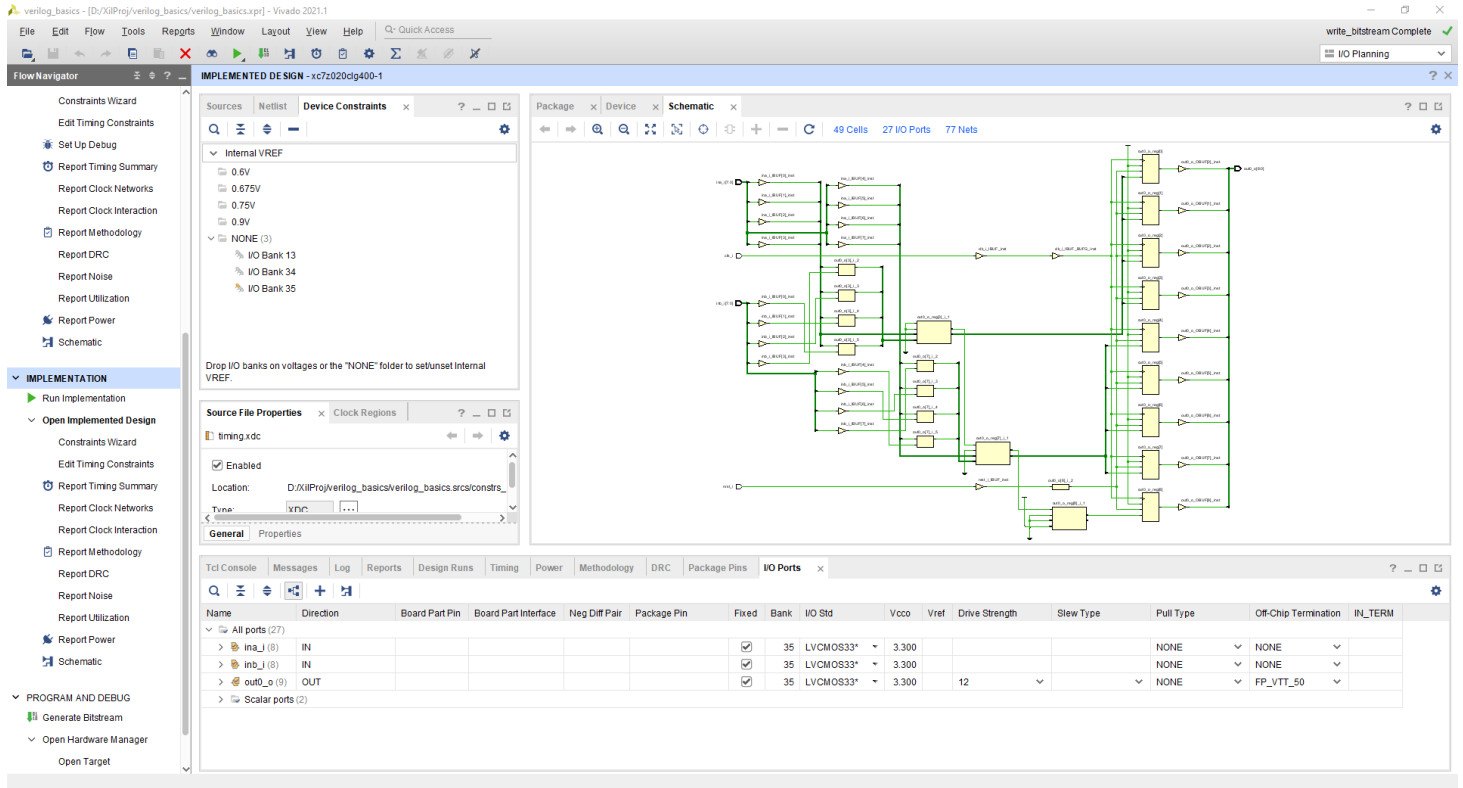
Design Flow

- You should be able to see the “Device” view now



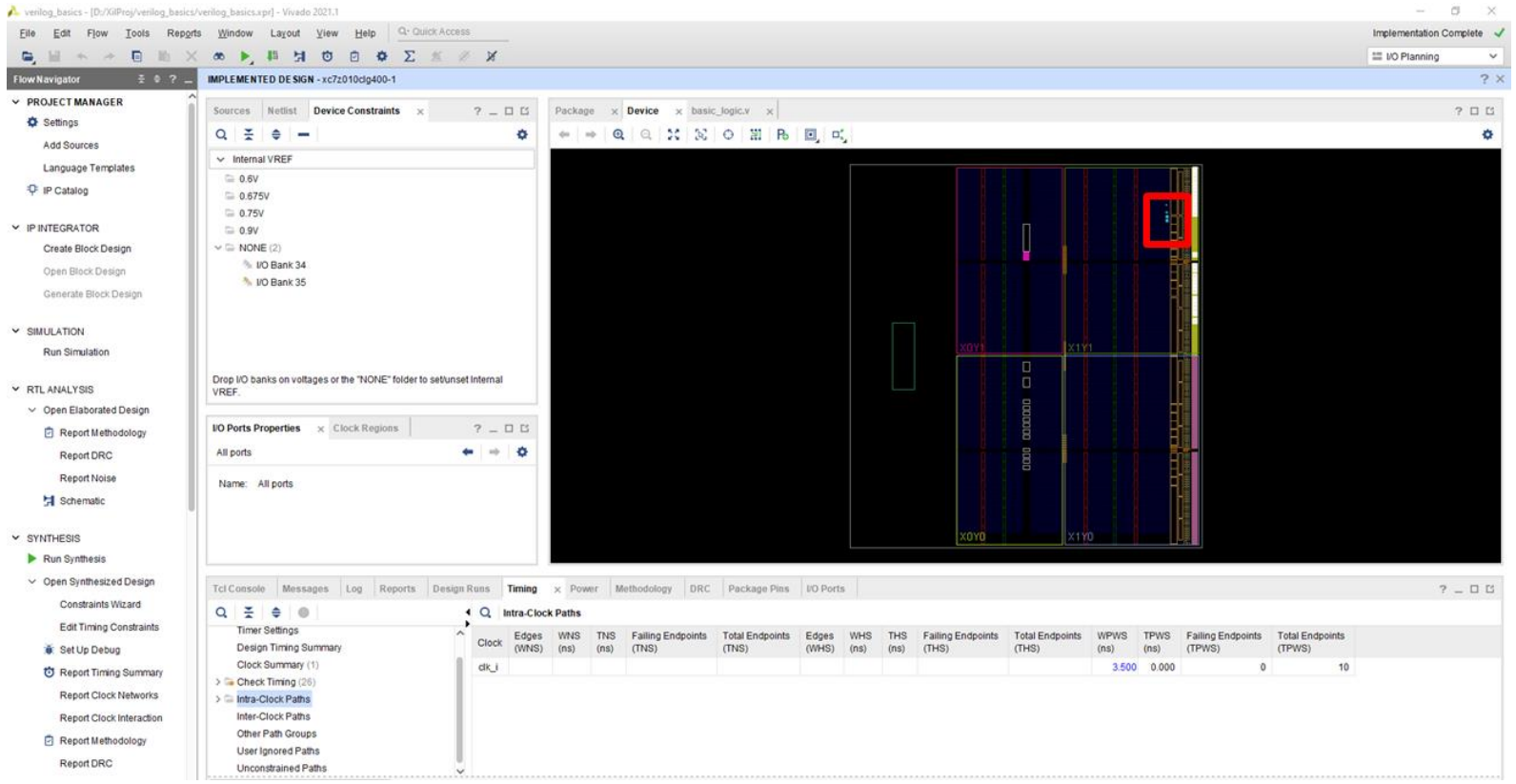
Design Flow

- Open the Schematic view to analyze the result from the implementation
- Play around with the schematic. Why is it so different from the schematic generated by the “Elaborated Design”?



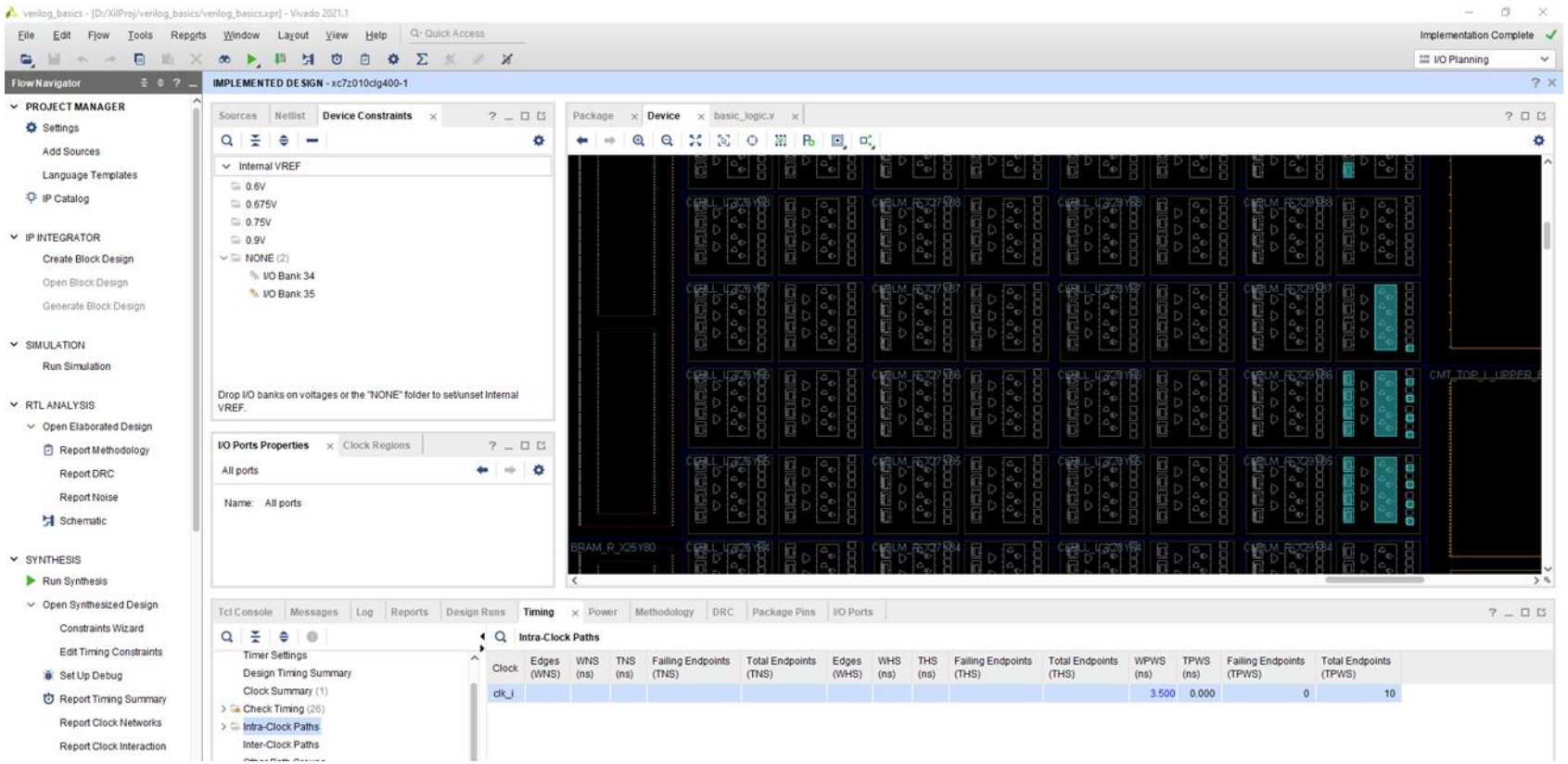
Design Flow

- Return to the “Device” view and zoom-in on the window by drawing a rectangle over the small blue dots (click and hold the left mouse button and drag to draw the rectangle)



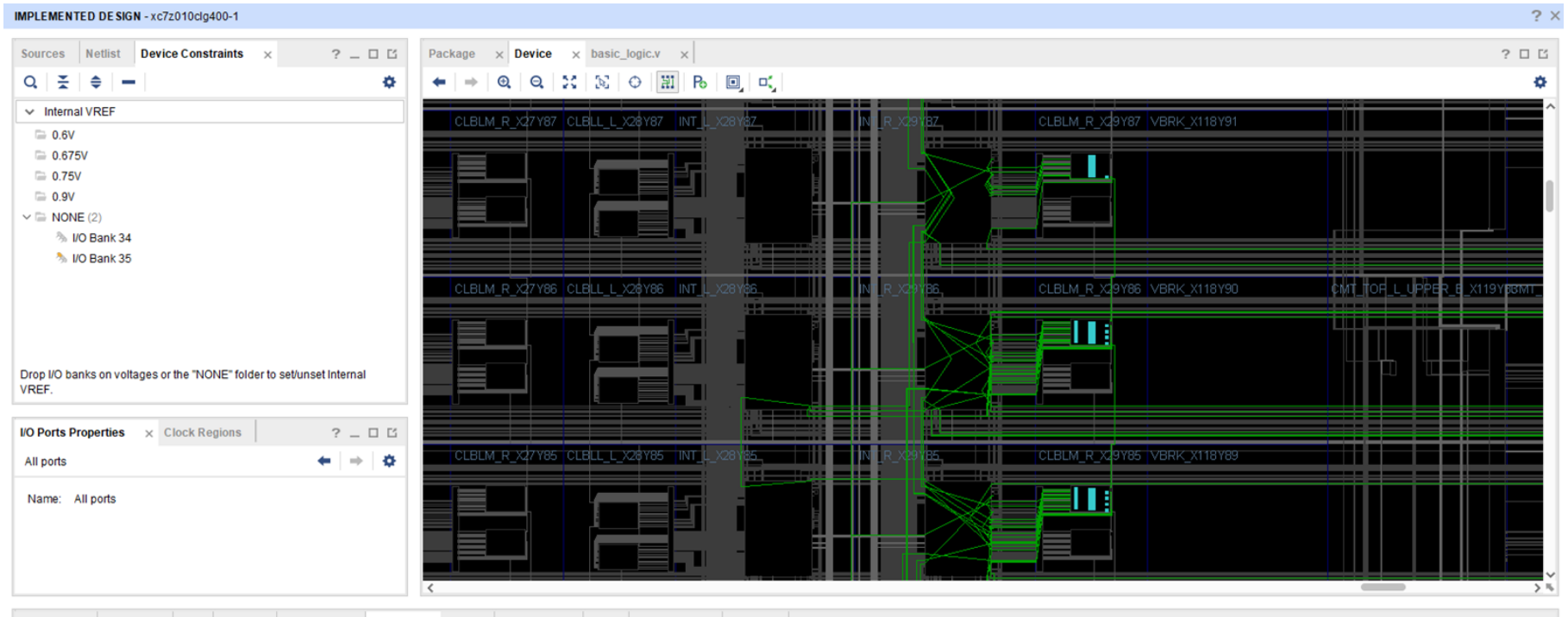
Design Flow

- What are those blue block?



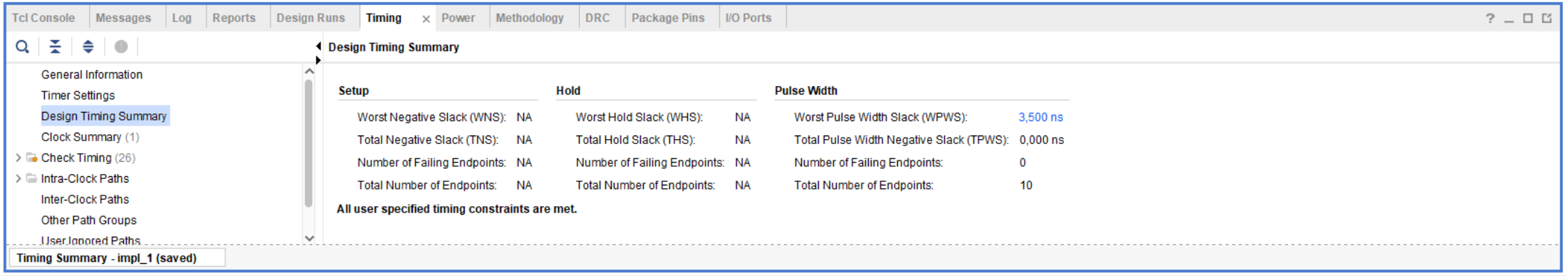
Design Flow

- Click on the “Routing resources icon”. What happened? Why is this view useful?
- Explore this view and once satisfied, click on the icon again to return to the default view



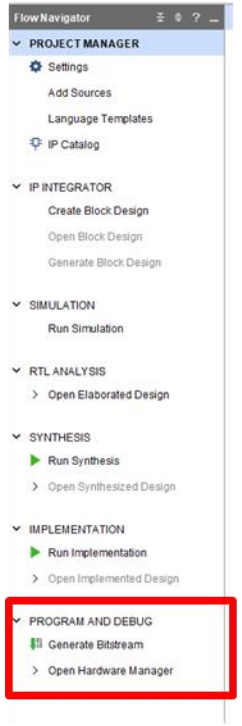
Design Flow

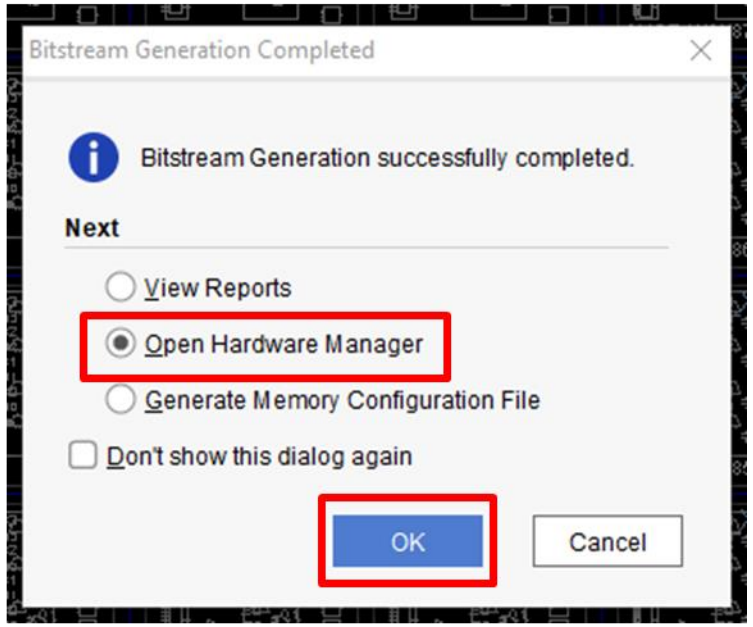
- Check the timing tab at the bottom of the window to see if the design pass timing



Design Flow

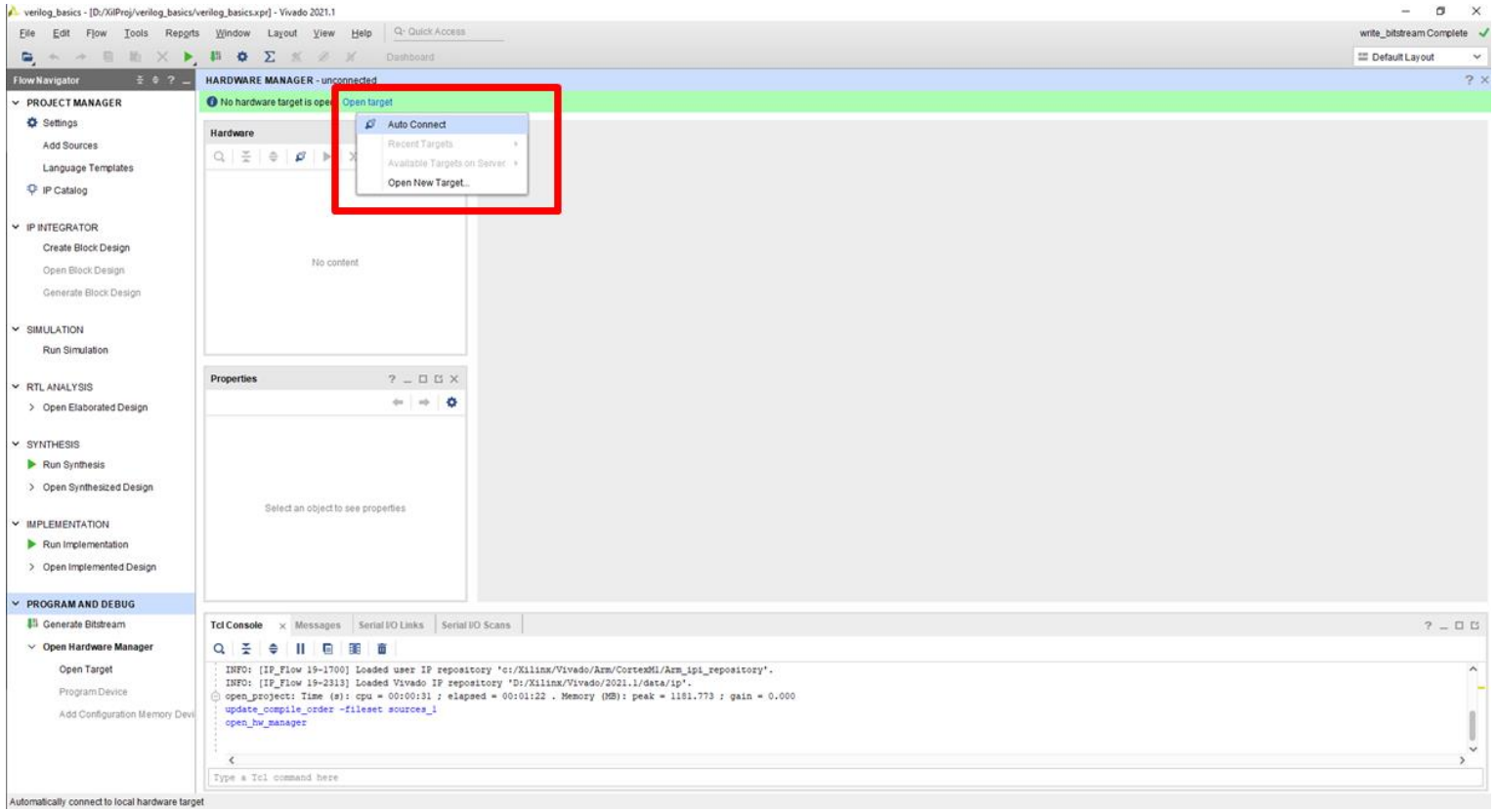
- Everything should be ok with our design, so it is time to generate the bitstream.
- Select “Generate Bitstream” and once the process is finished select “Open Hardware manager” and then “OK”





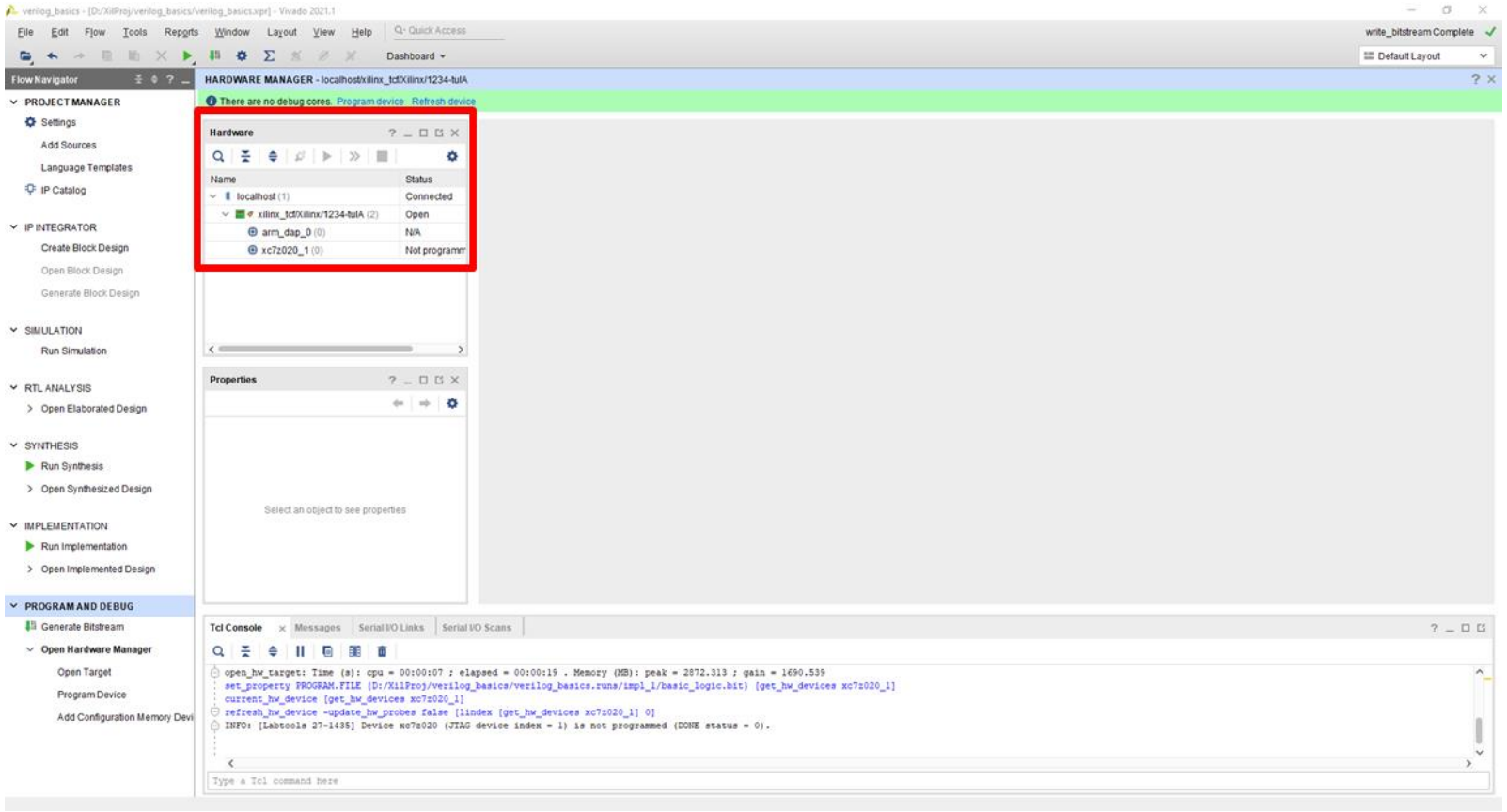
Design Flow

- On the hardware manager select “Open target” and then “Auto Connect” (make sure you have your board properly connected to your PC and that the board is powered up before doing this)



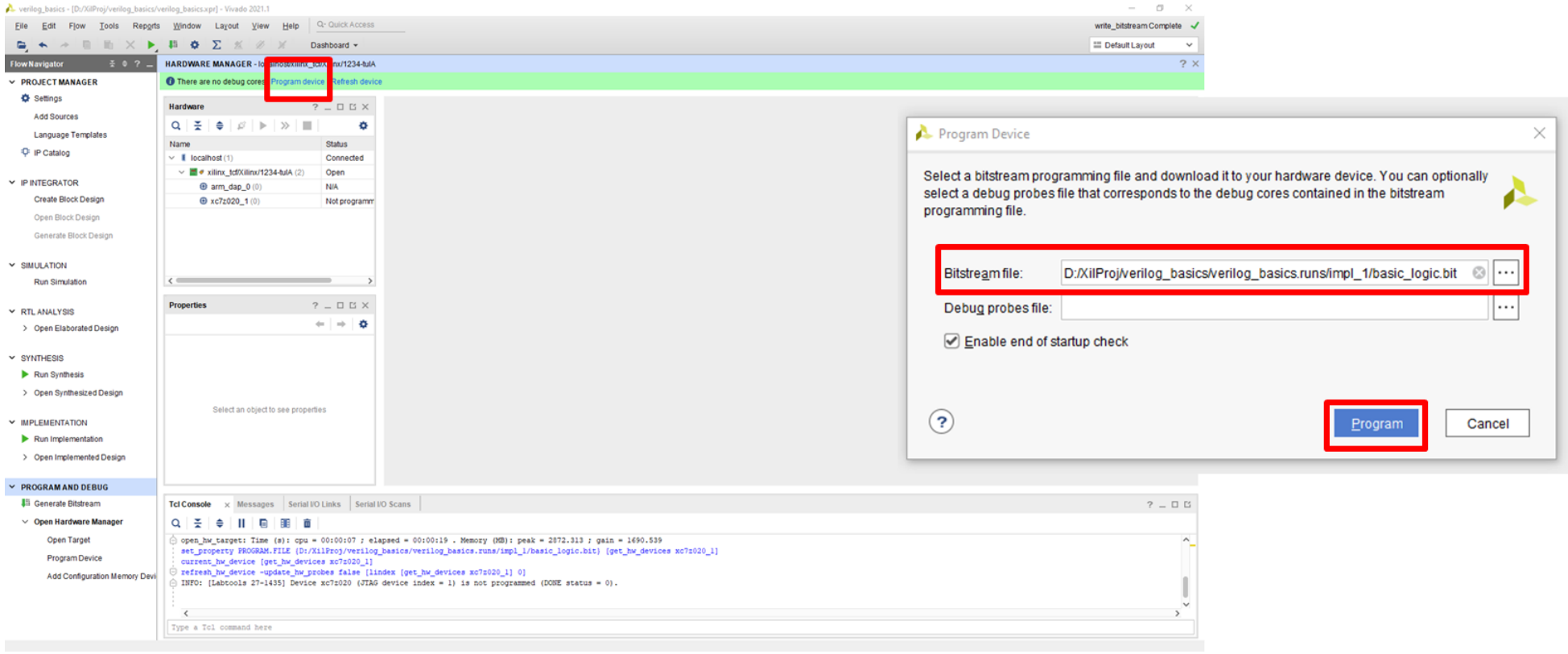
Design Flow

- You should be able to see your device with an Arm core (PS) and an FPGA IC (PL)



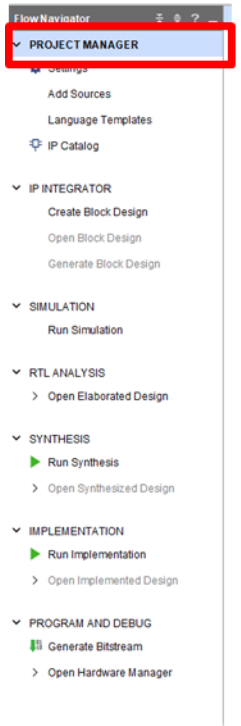
Design Flow

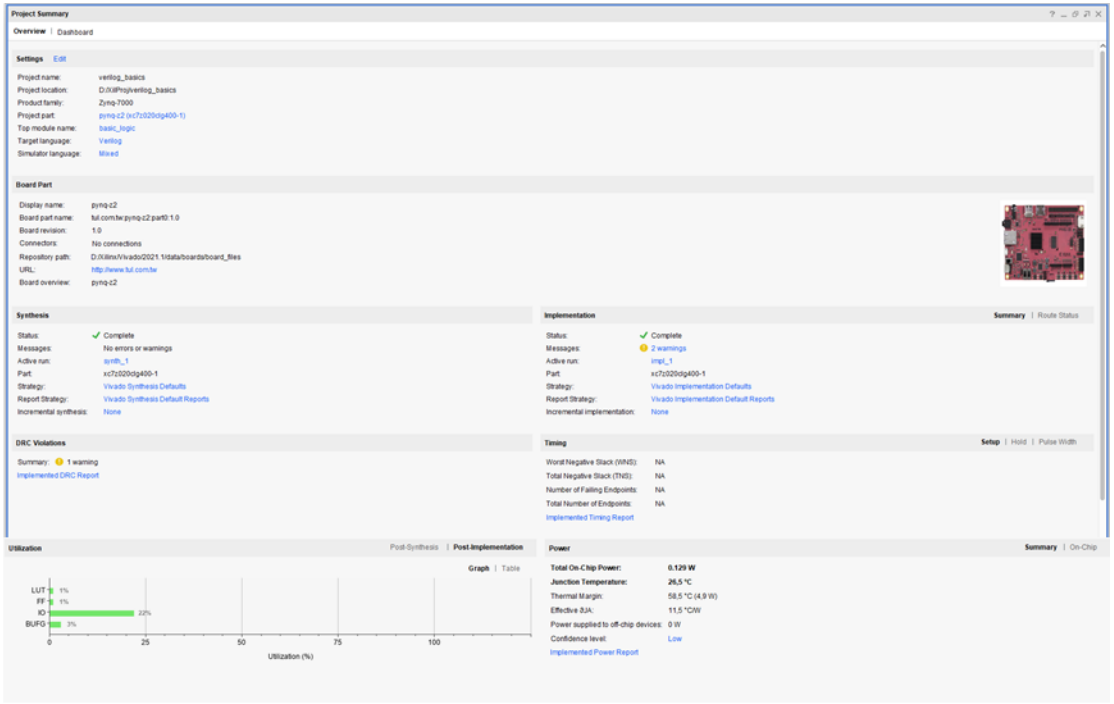
- Select “Program device” and on the next window select the proper bitstream file (it should be automatically filled in by the tool). Click “Program” to download the bitstream to the board



Design Flow

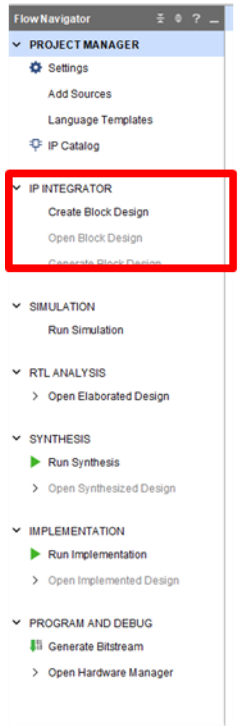
- Congratulations! Your first design is up and running on real hardware
- Let's have a look at our project overview after finishing all the design steps. Click on “Project Manager”. What relevant information can you see there? Why is it important?

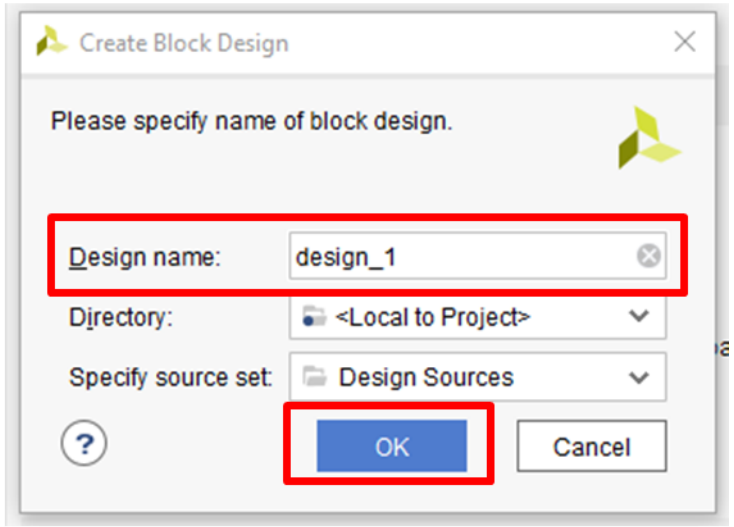




Design Flow

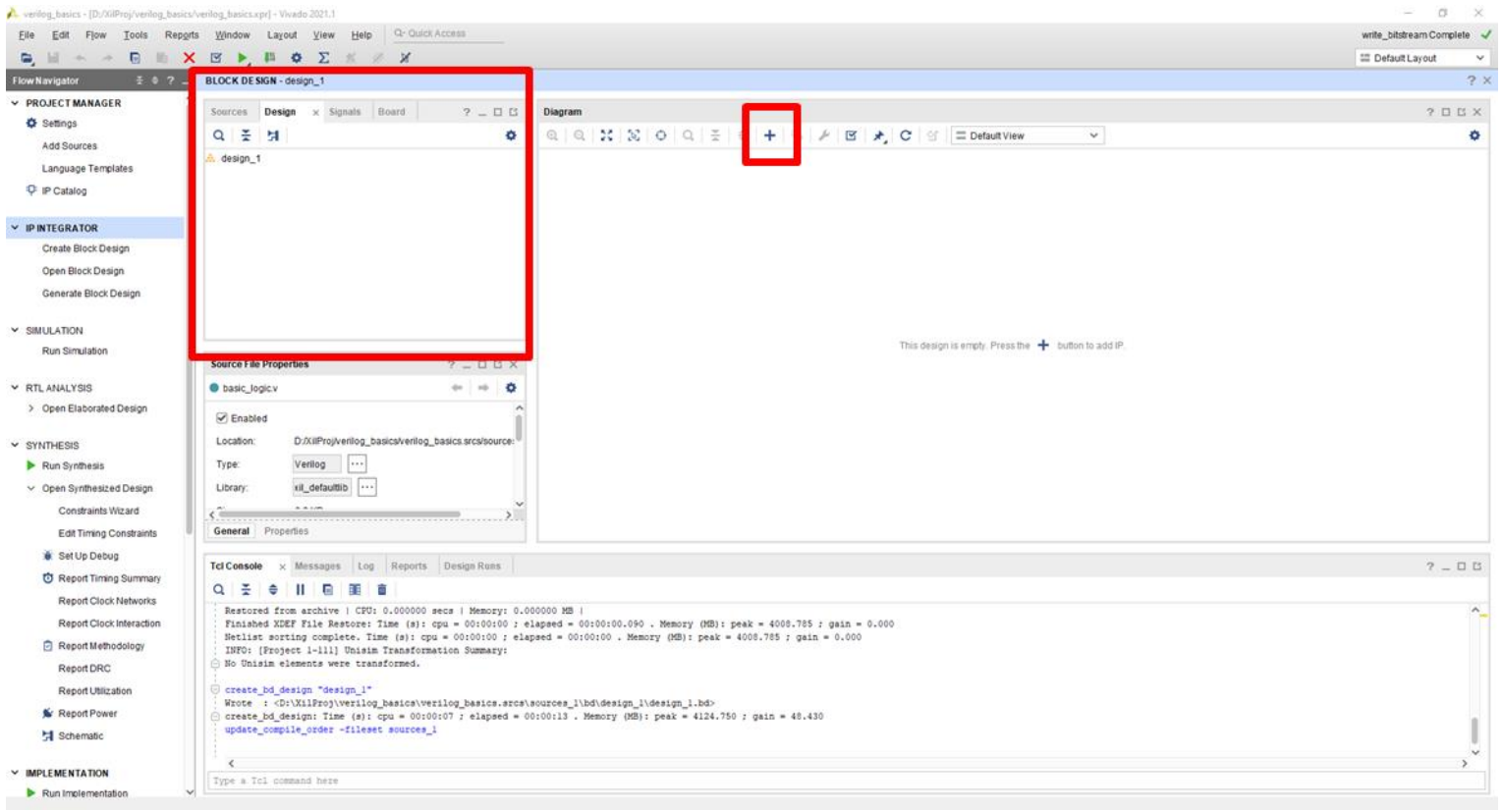
- The block design allows you to build more complex designs. Create a new block design by selecting “Create block Design” on the “IP Integrator”
- Chose a name for the design and click “OK”





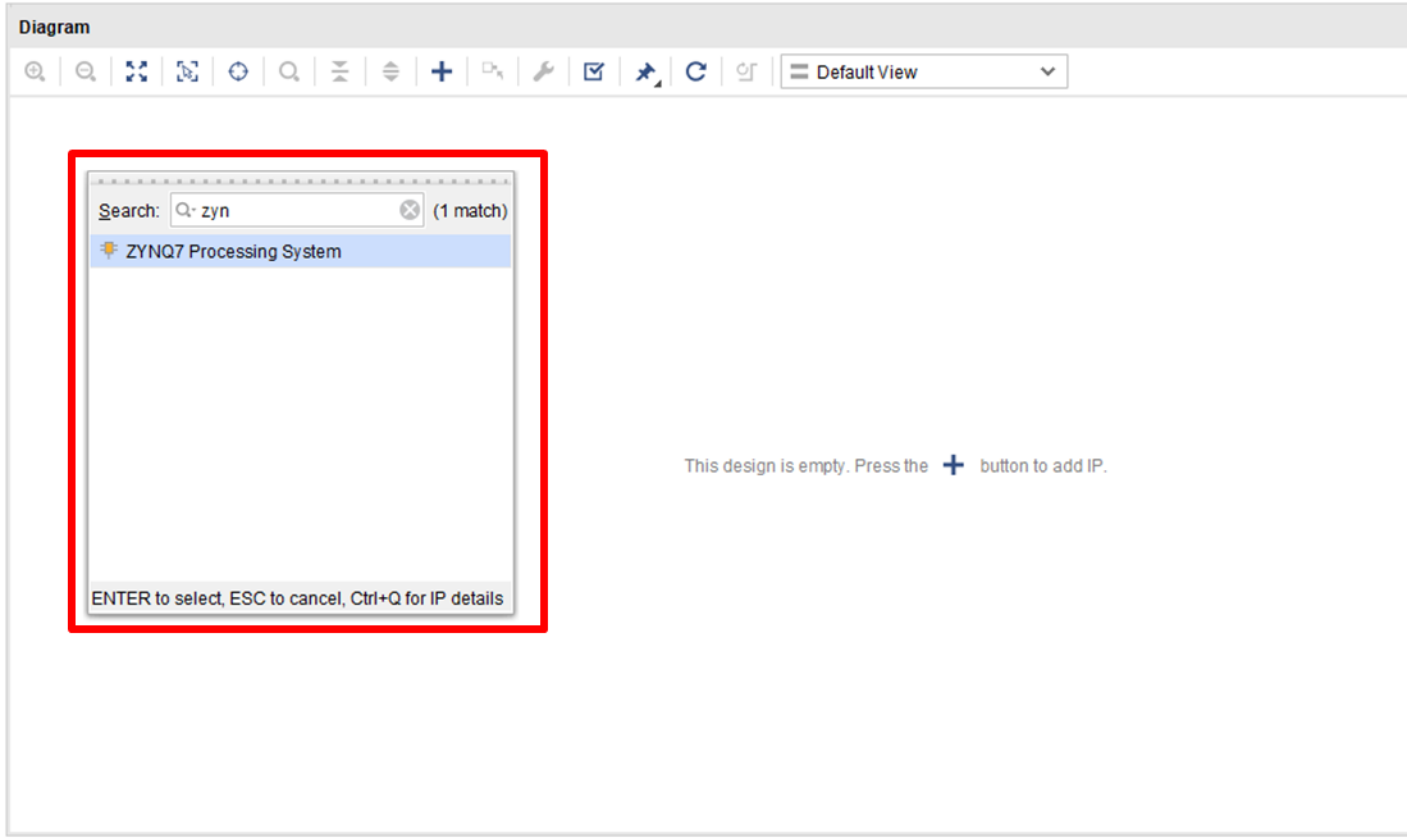
Design Flow

- On the Block design notice the new design on the left
- Click the “+” sign to add a new IP from the libraries available



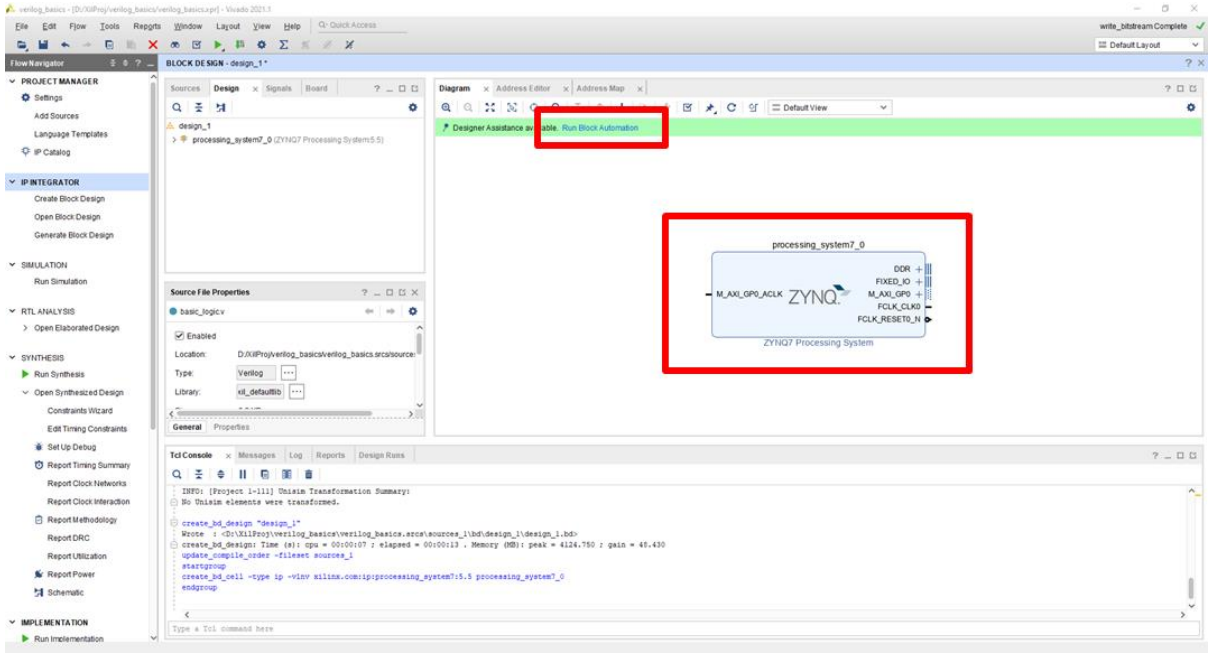
Design Flow

- Type “zynq” on the search bar and double click on the “ZYNQ Processing System” to add it to the design



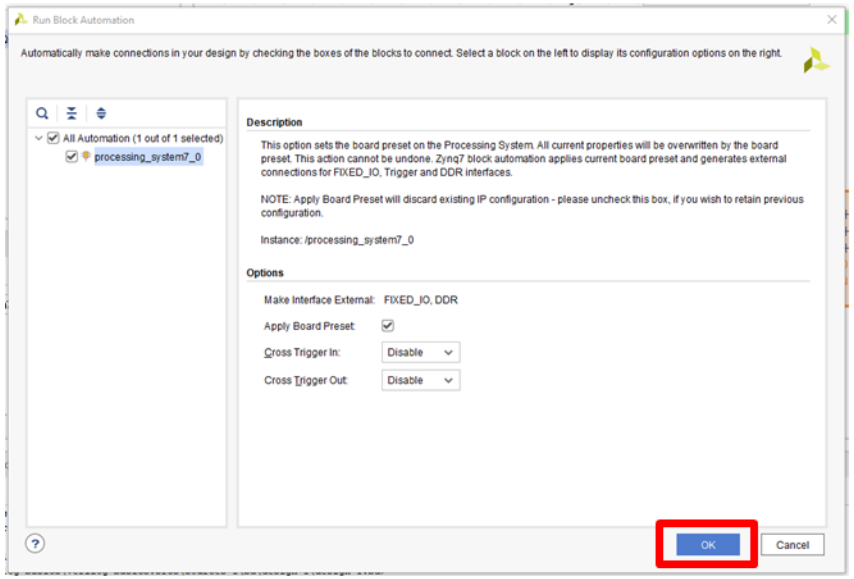
Design Flow

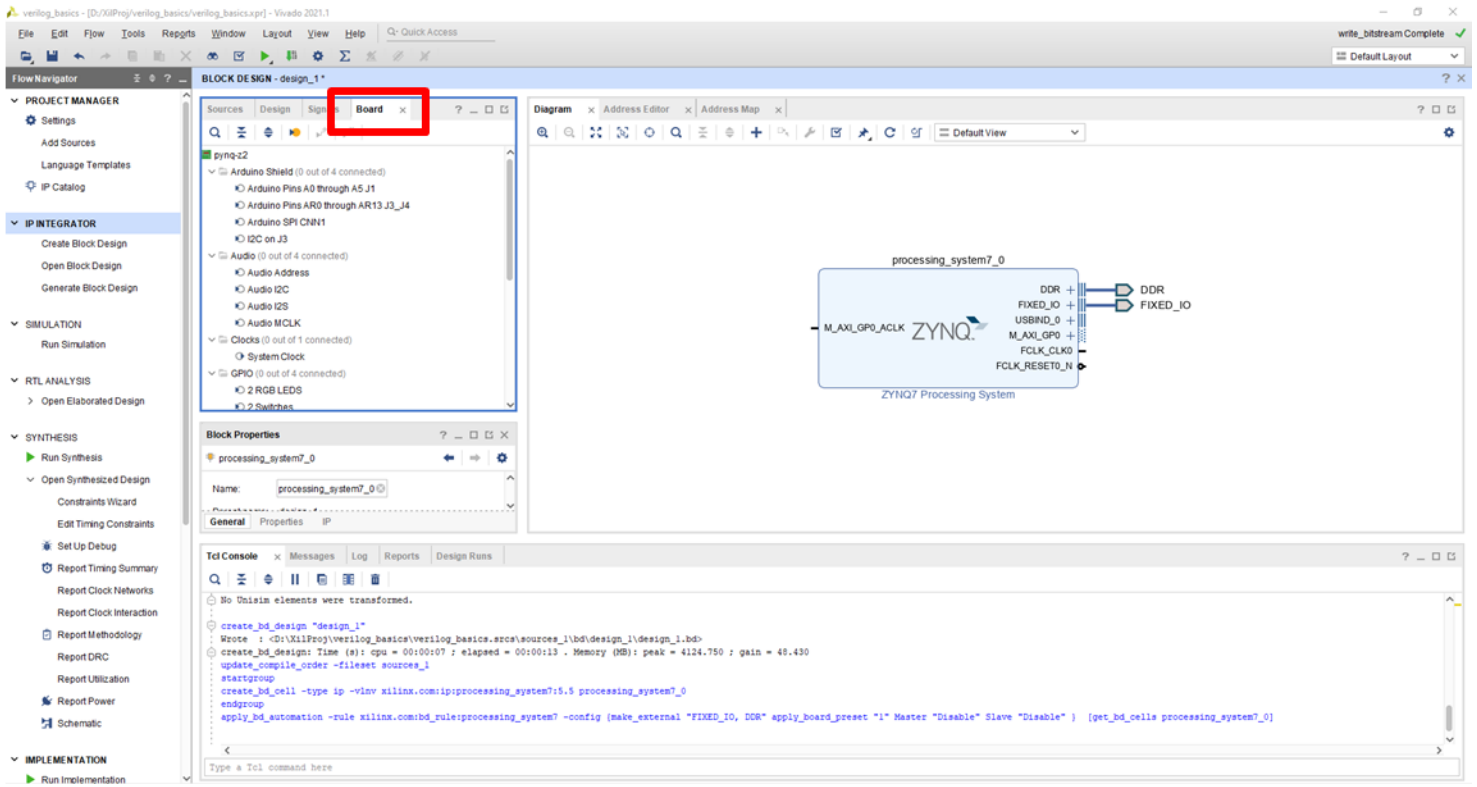
- Now we have the Arm A9 core as part of our design.
- This is a Hard core included in the ZYNQ SoC we are using. Depending on the board you are using you may or may not have a hard core, and you might as well have a different hardcore or a multicore
- Click on “Run Block Automation”



Design Flow

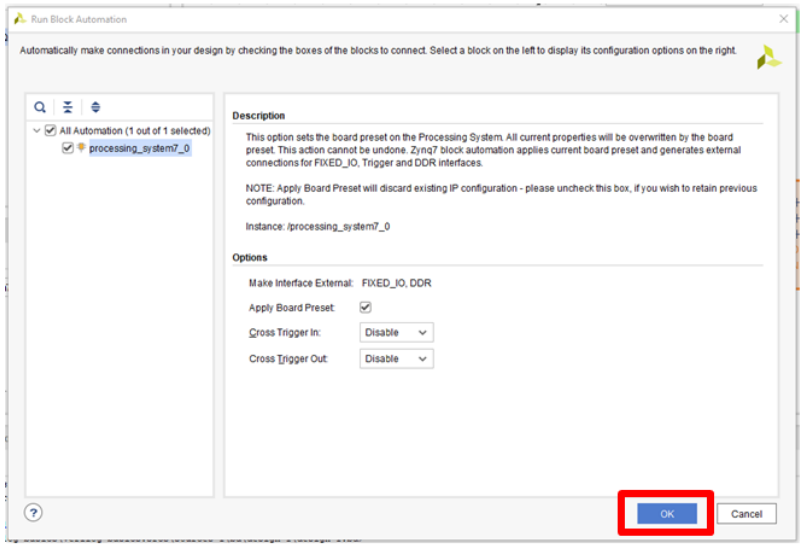
- Click “OK”. What changed in the design?
- Change to the Board tab

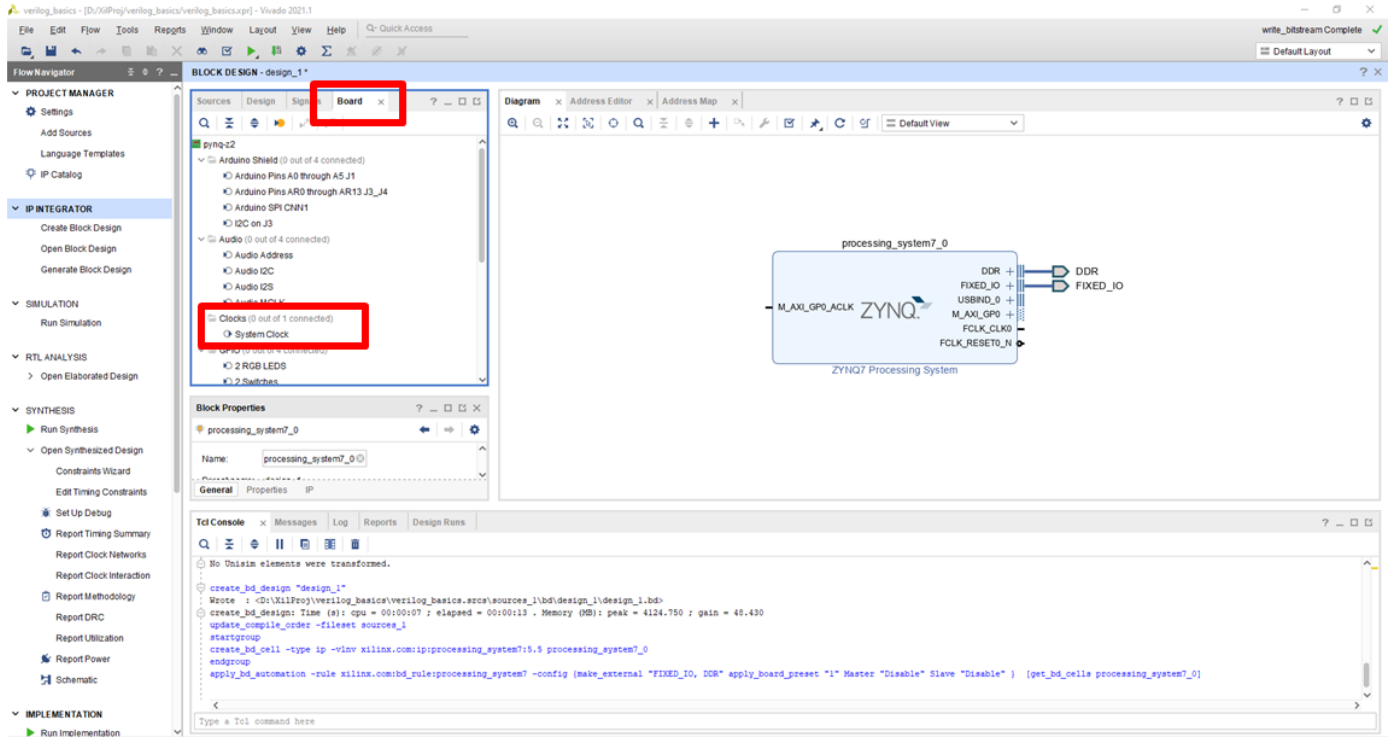




Design Flow

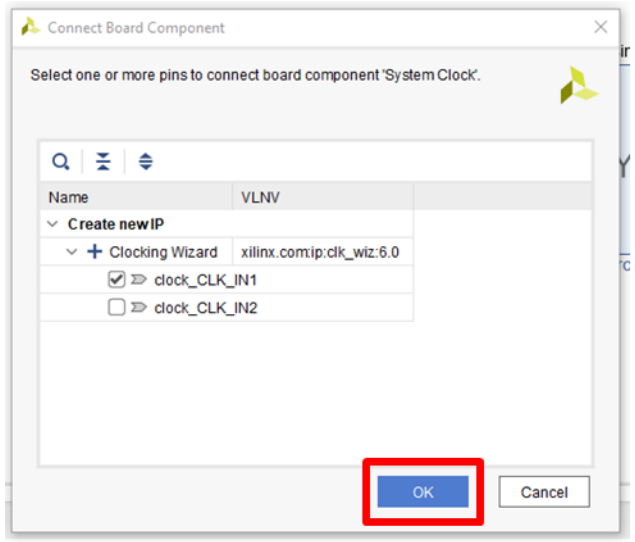
- Click “OK”. What changed in the design?
- Change to the Board tab and double click, or drag & drop into the Diagram window, the “System Clock”

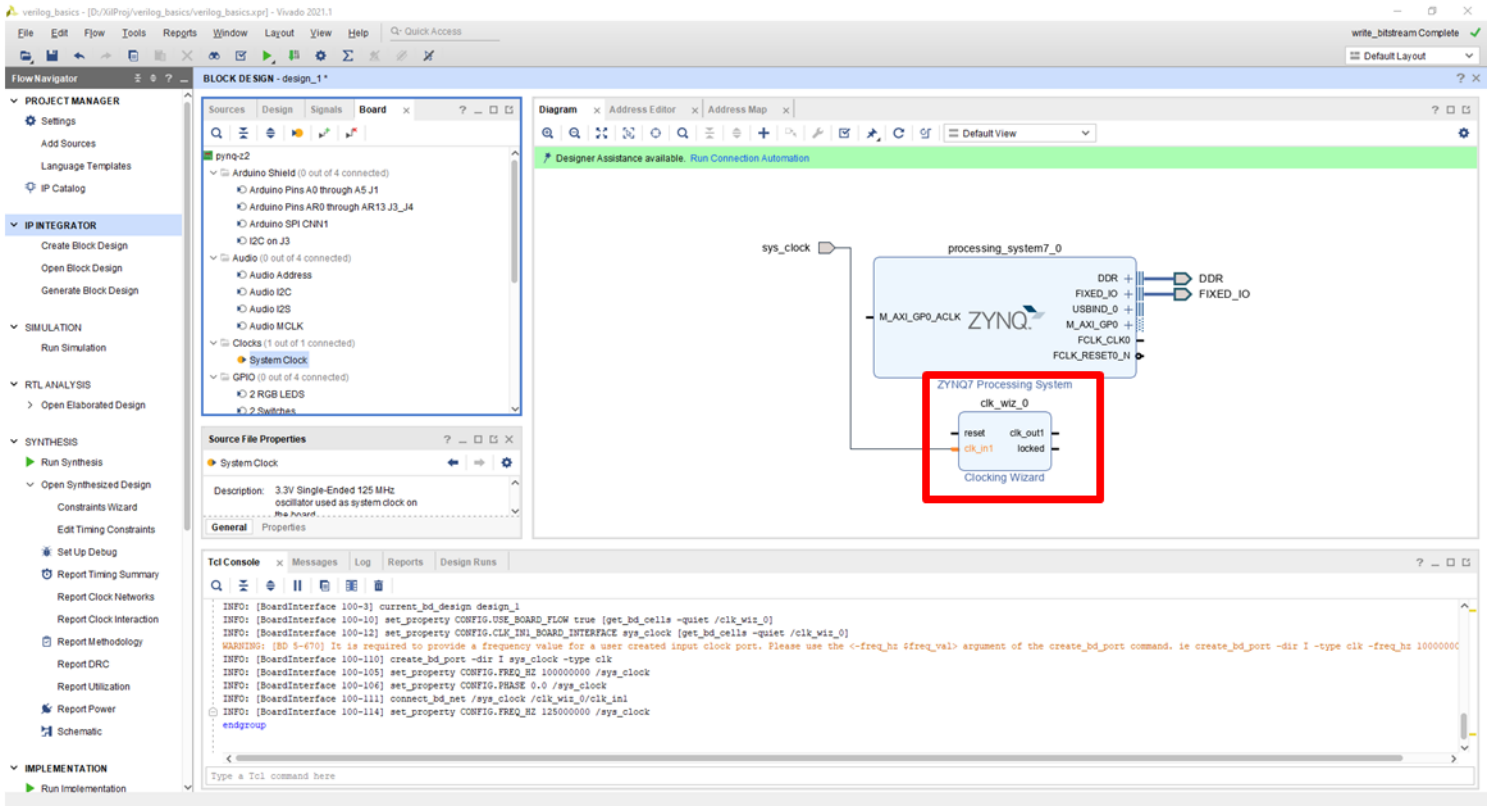




Design Flow

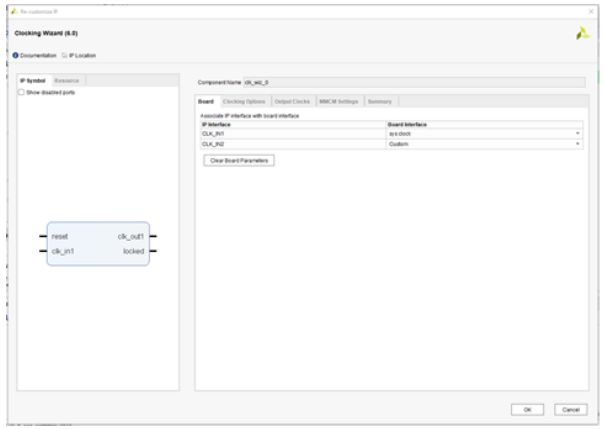
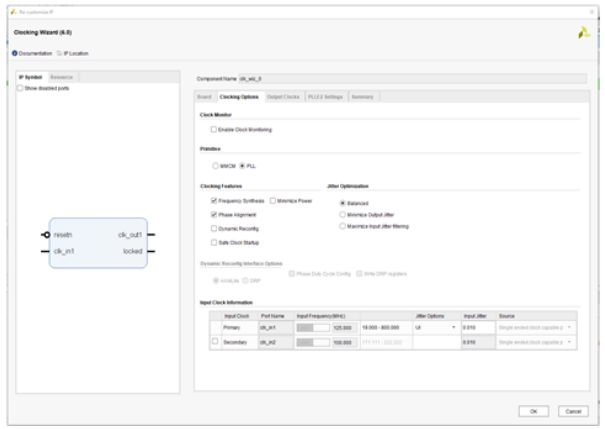
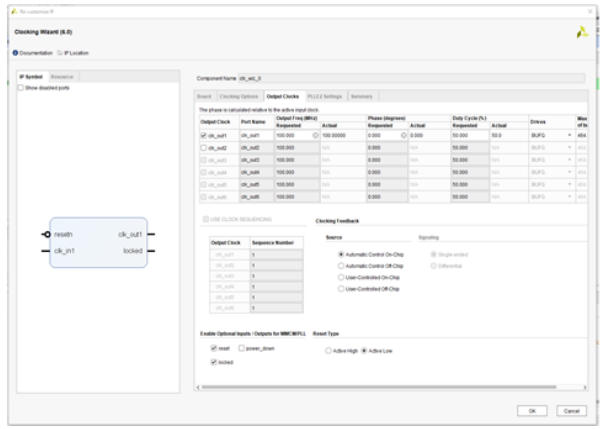
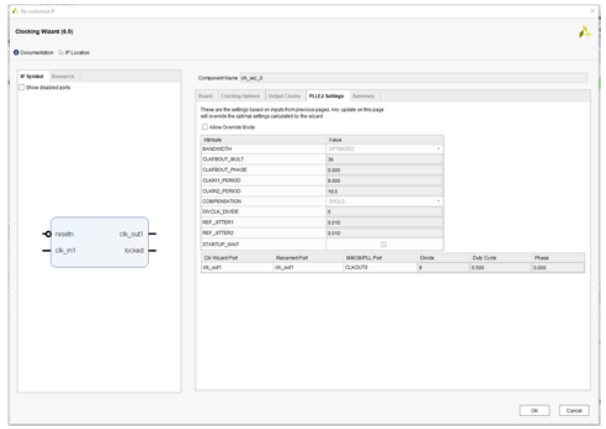
- Click “OK” to validate the configurations
- Double click on the “clk_wiz_0” module to configure it





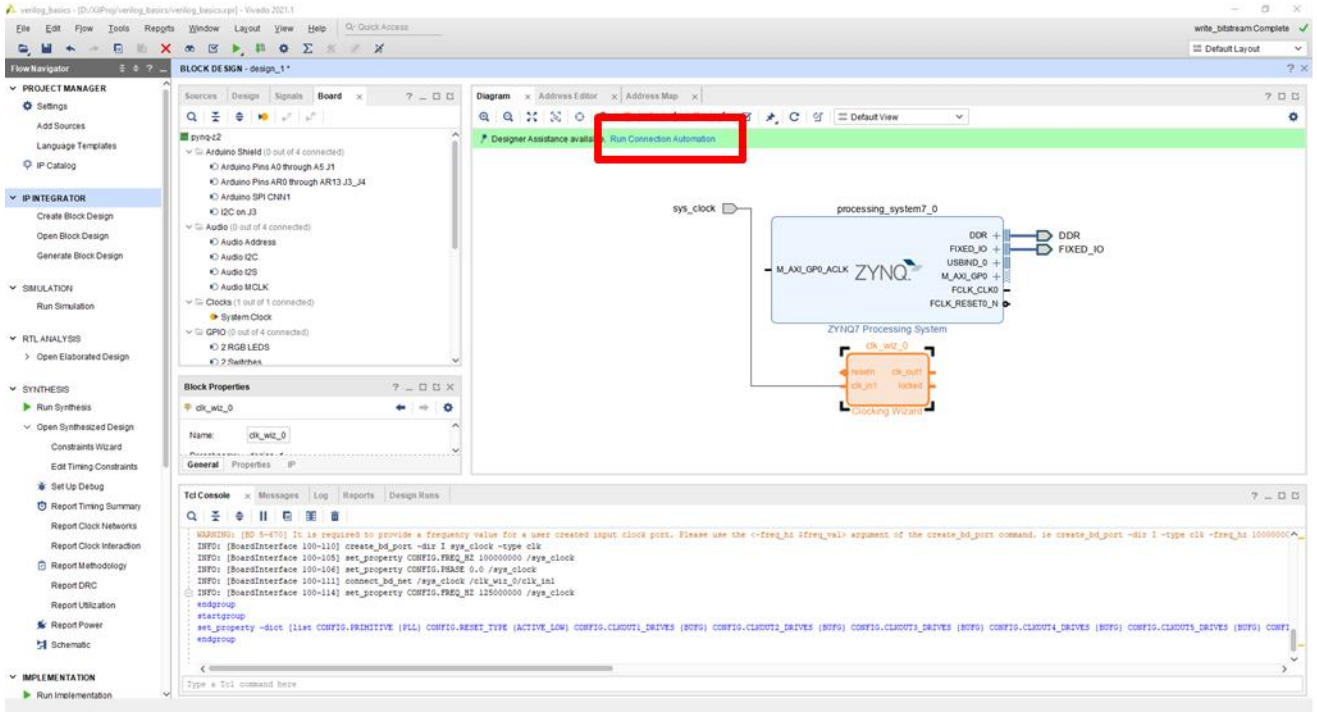
Design Flow

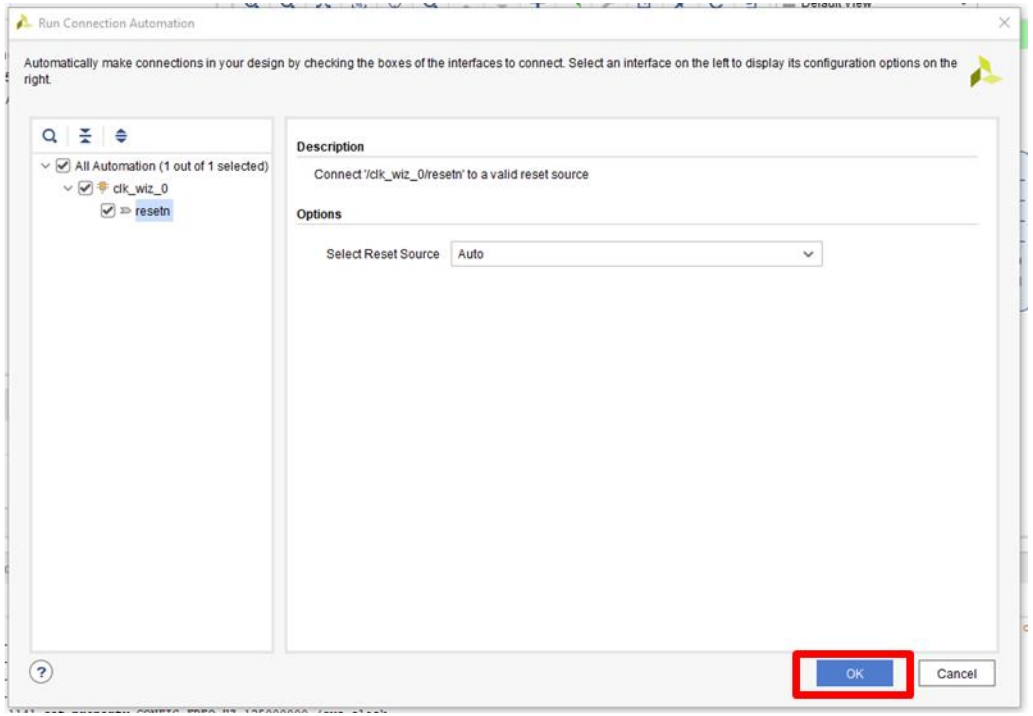
- Explore the different configuration tabs
- Make sure “PLL” is selected on the second tab and that the “Reset Type” is set to “Active Low” on the third tab. Left the other options as default. Why did you need to set the reset to active low?
- Click Ok to finish the configuration

Design Flow

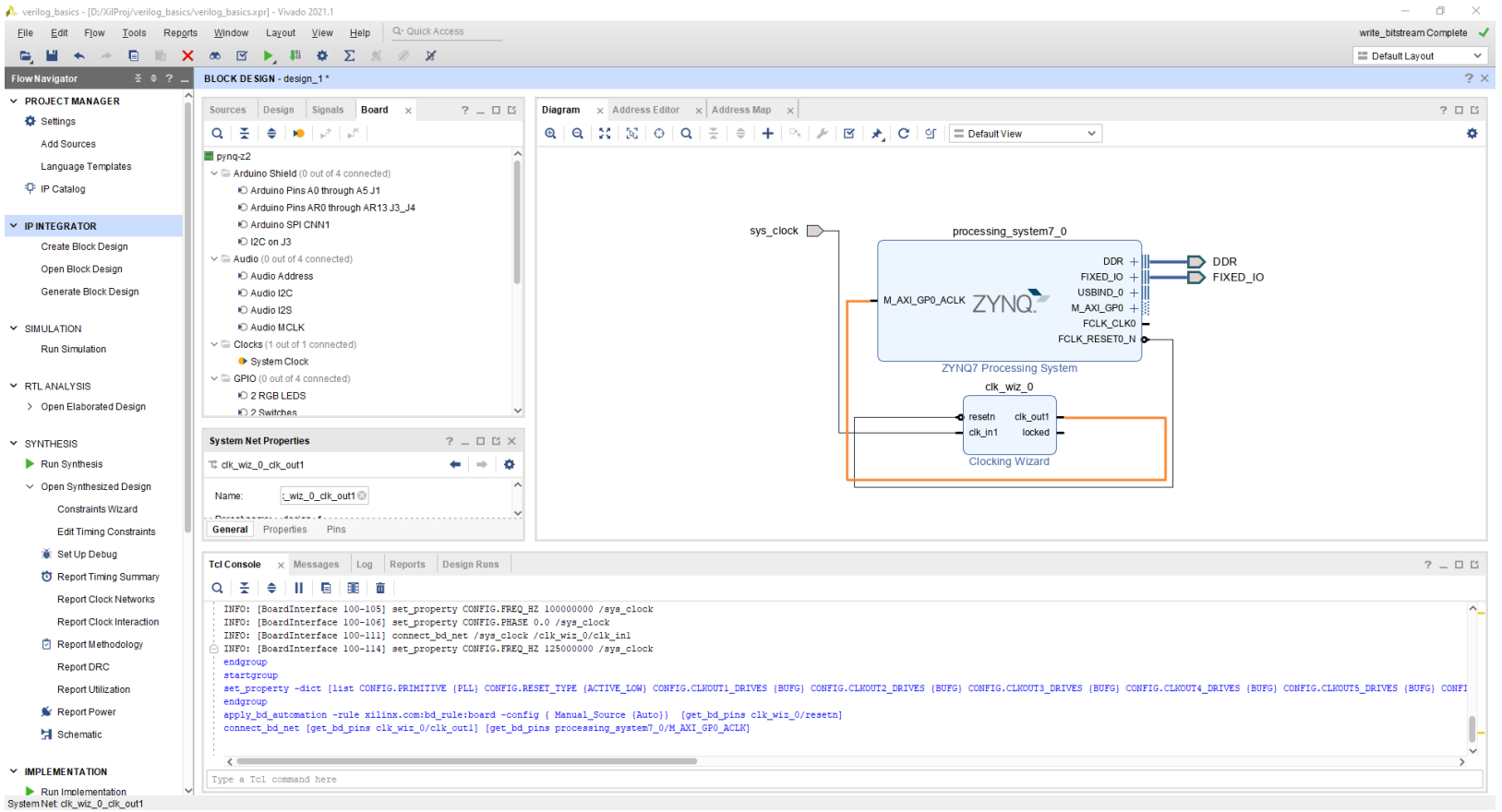
- Click on “Run Connection Automation” and then click “OK” to confirm the default connections





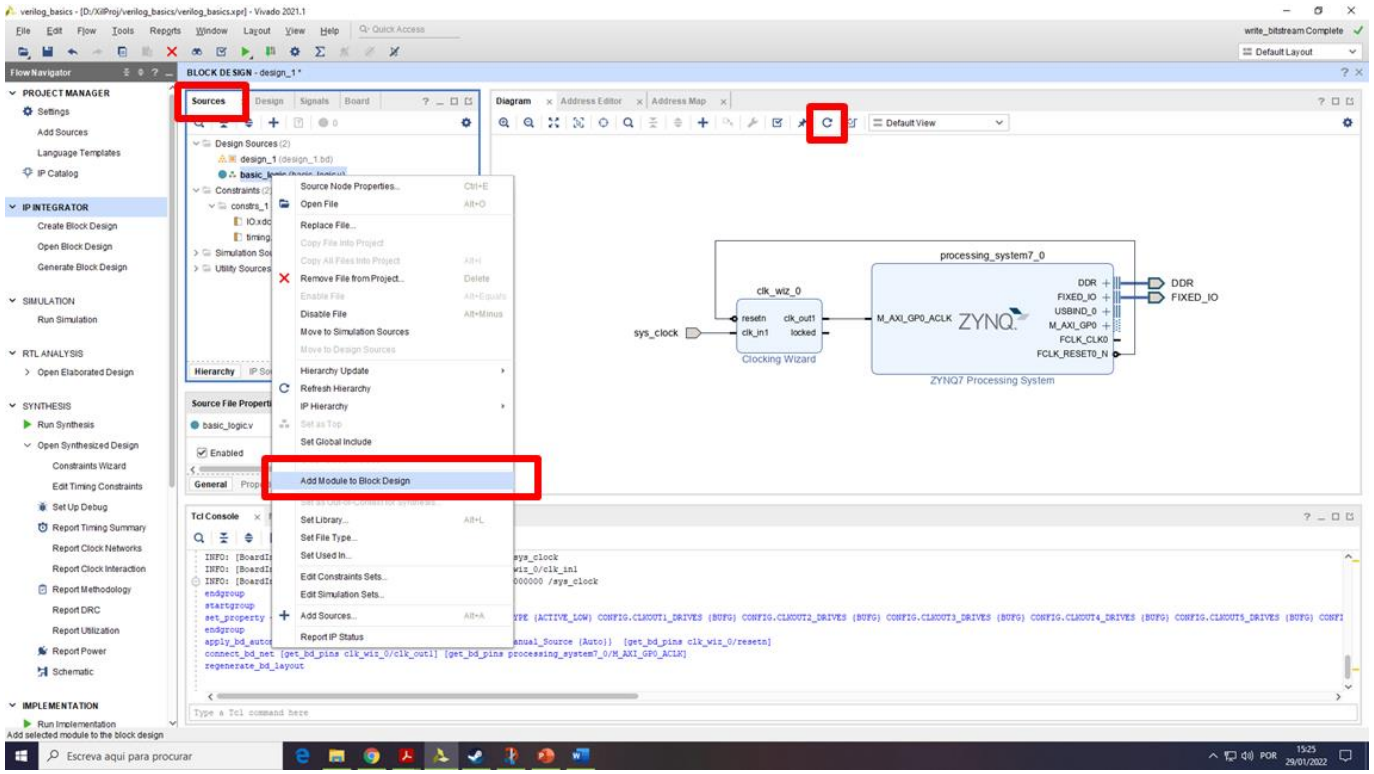
Design Flow

- Connect the “clock_out1” output from the “clk_wiz_0” to the “M_AXI_CP0_ACLK” input of the ZYNQ module



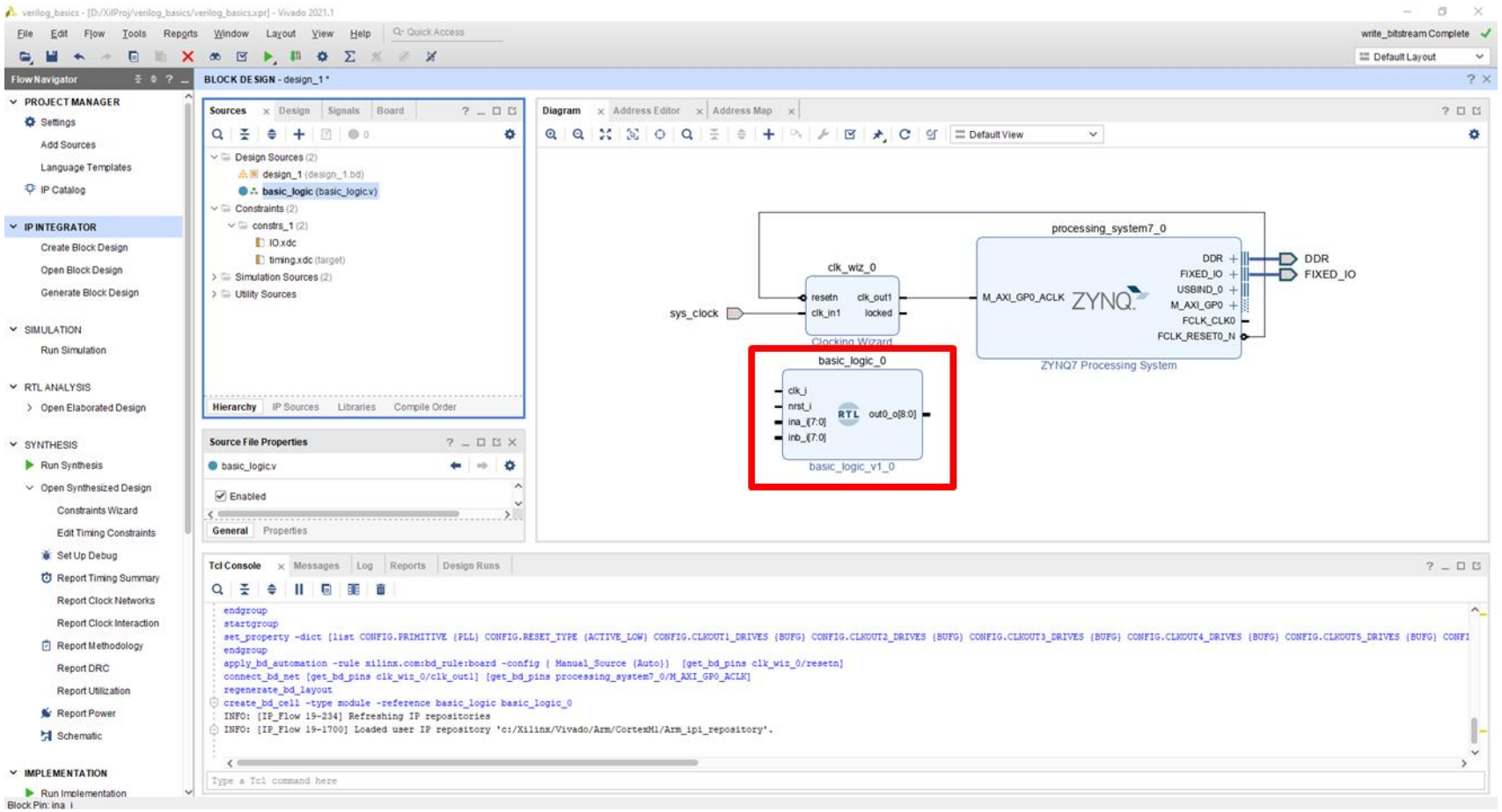
Design Flow

- You can rearrange the design by clicking on the “Regenerate Layout” icon.
- Let's add our module to the design. On the Sources tab, right-click our module and select “Add Module to Block Design”



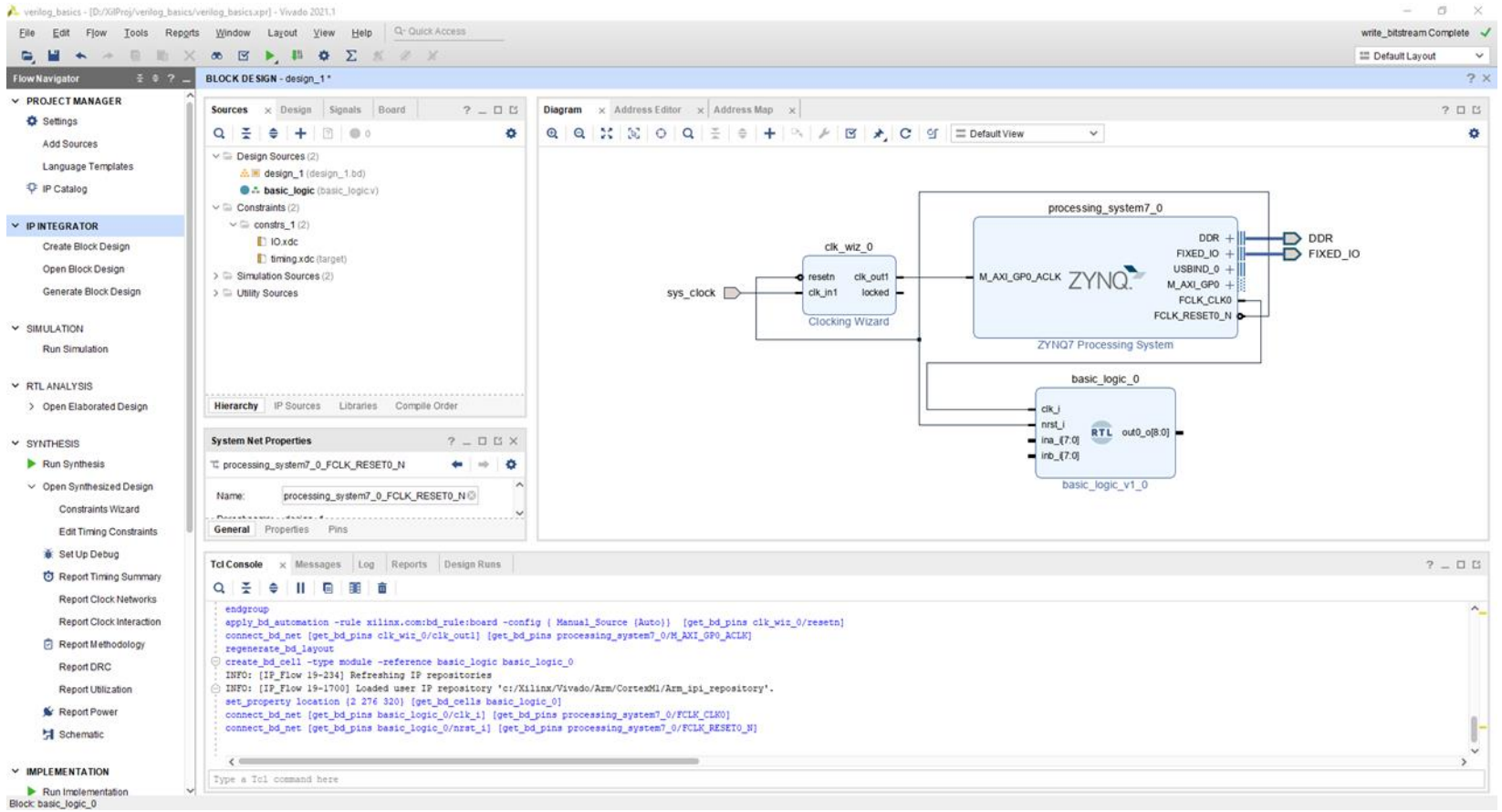
Design Flow

- Our module is now part of the Block Design. Let's connect it



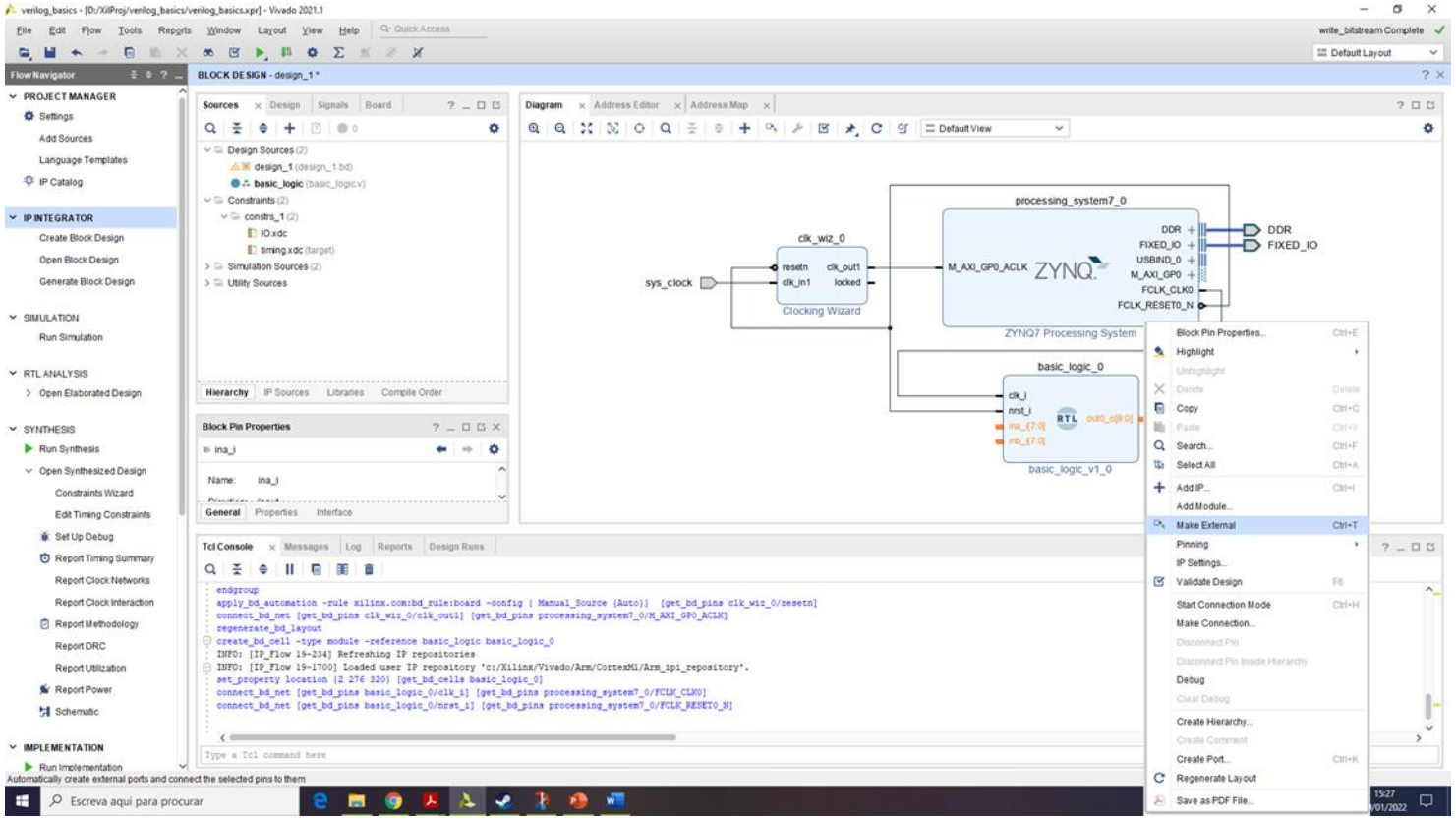
Design Flow

- Connect the clock and reset signals as shown in the figure



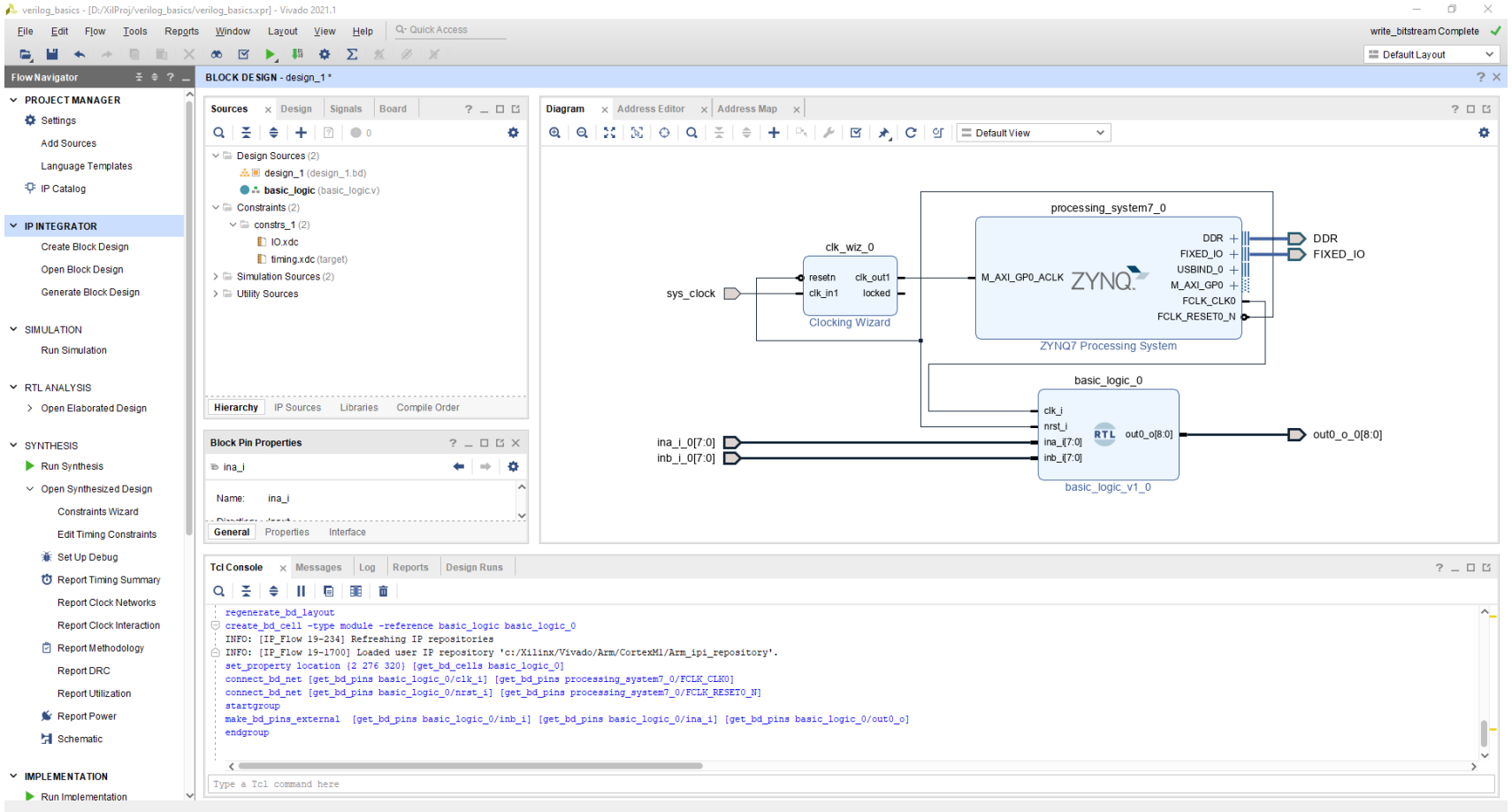
Design Flow

- Now select the ina, inb and out0_o ports (Select the first port and then hold control and left-click the other ports). Right-click and select “Make External” to create the ports’ interfaces.



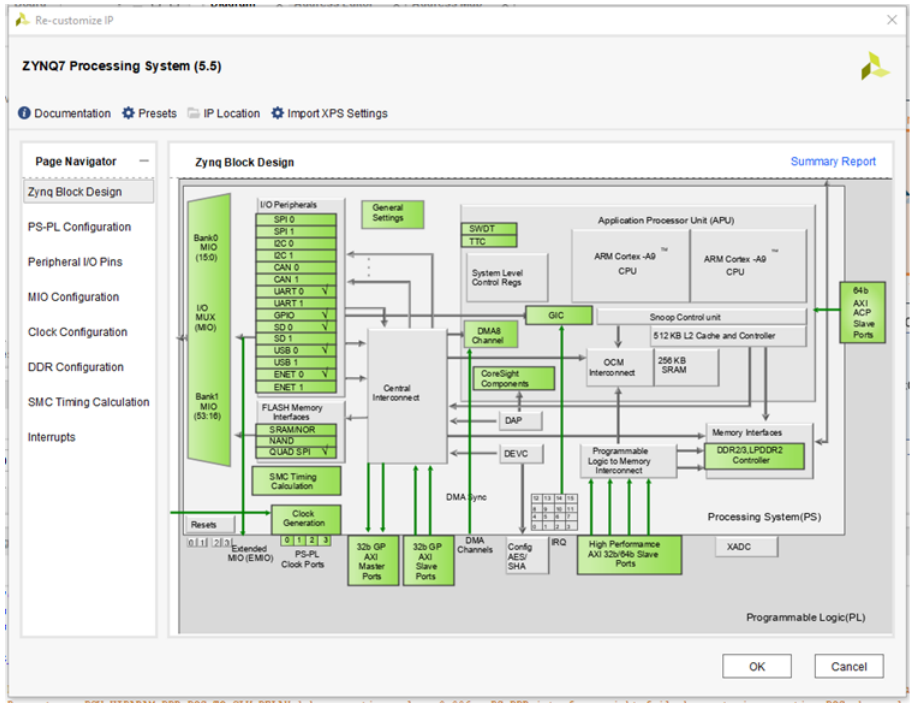
Design Flow

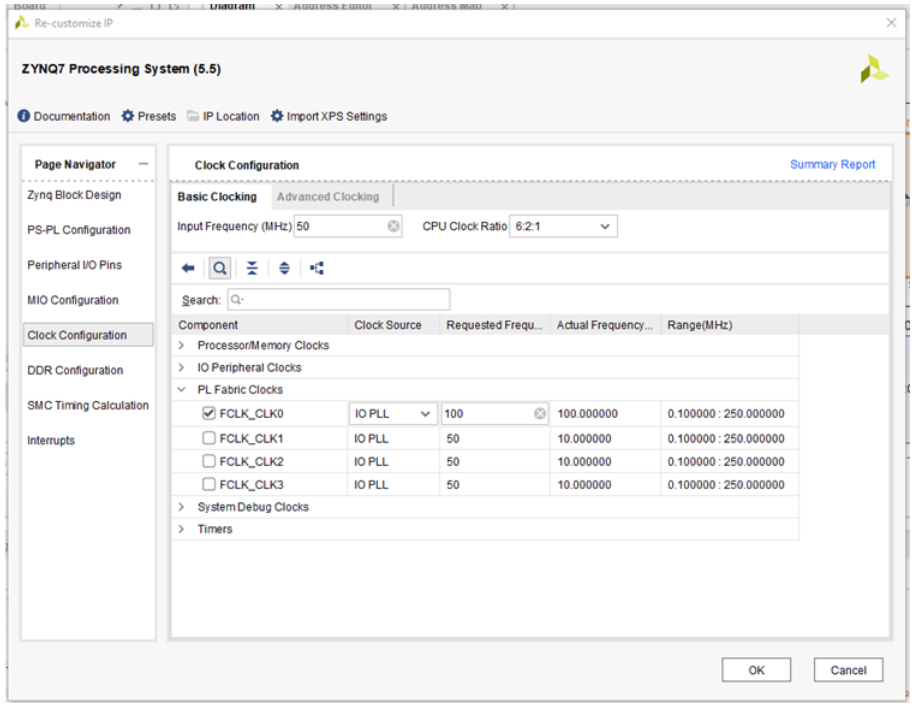
- The expected result



Design Flow

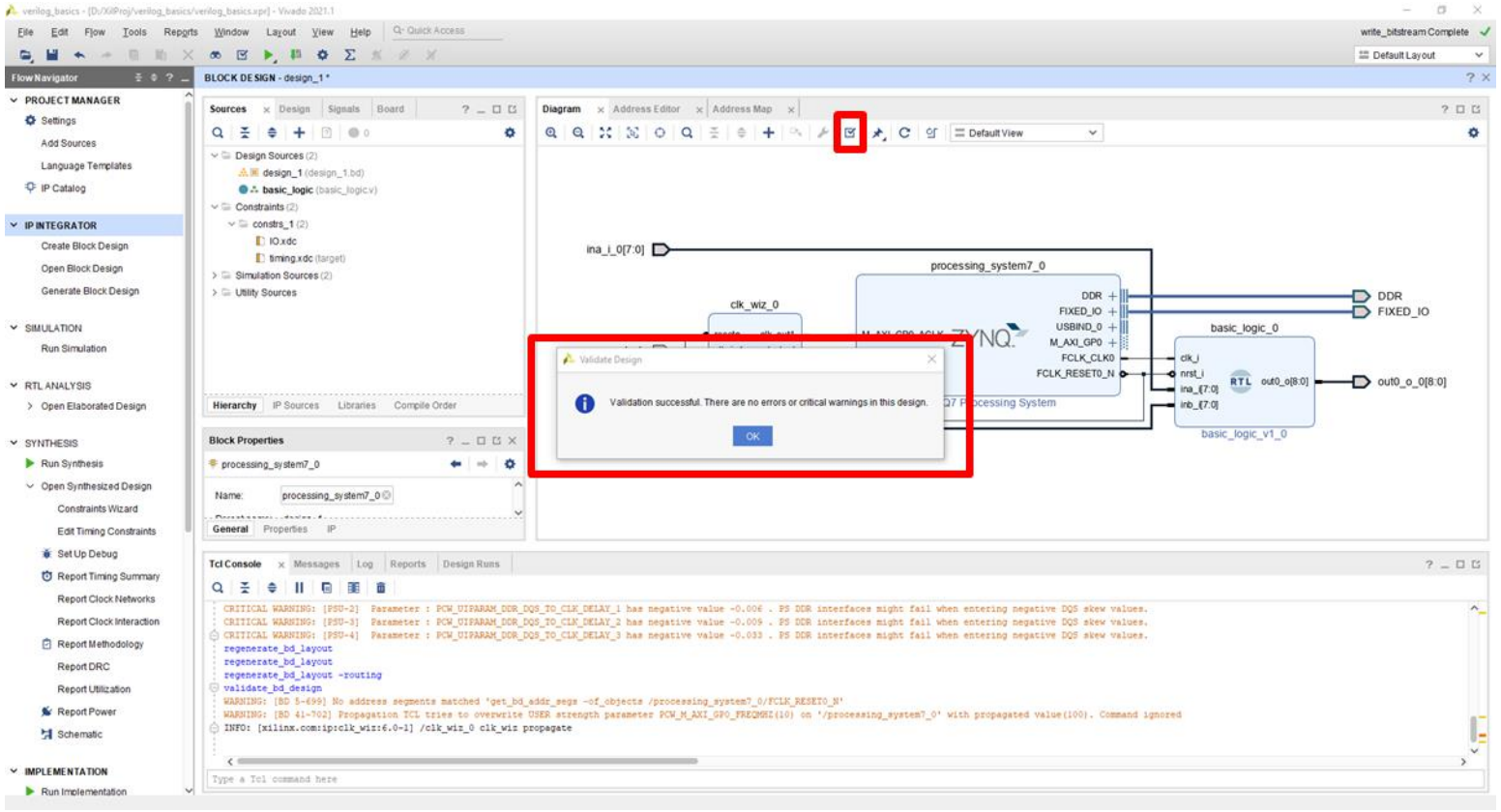
- Double-click on the ZYNQ block and explore the menus. Which parameters are available for configuration? Which is the frequency that is feeding our RTL module?
- Cancel once you finish exploring all the tabs





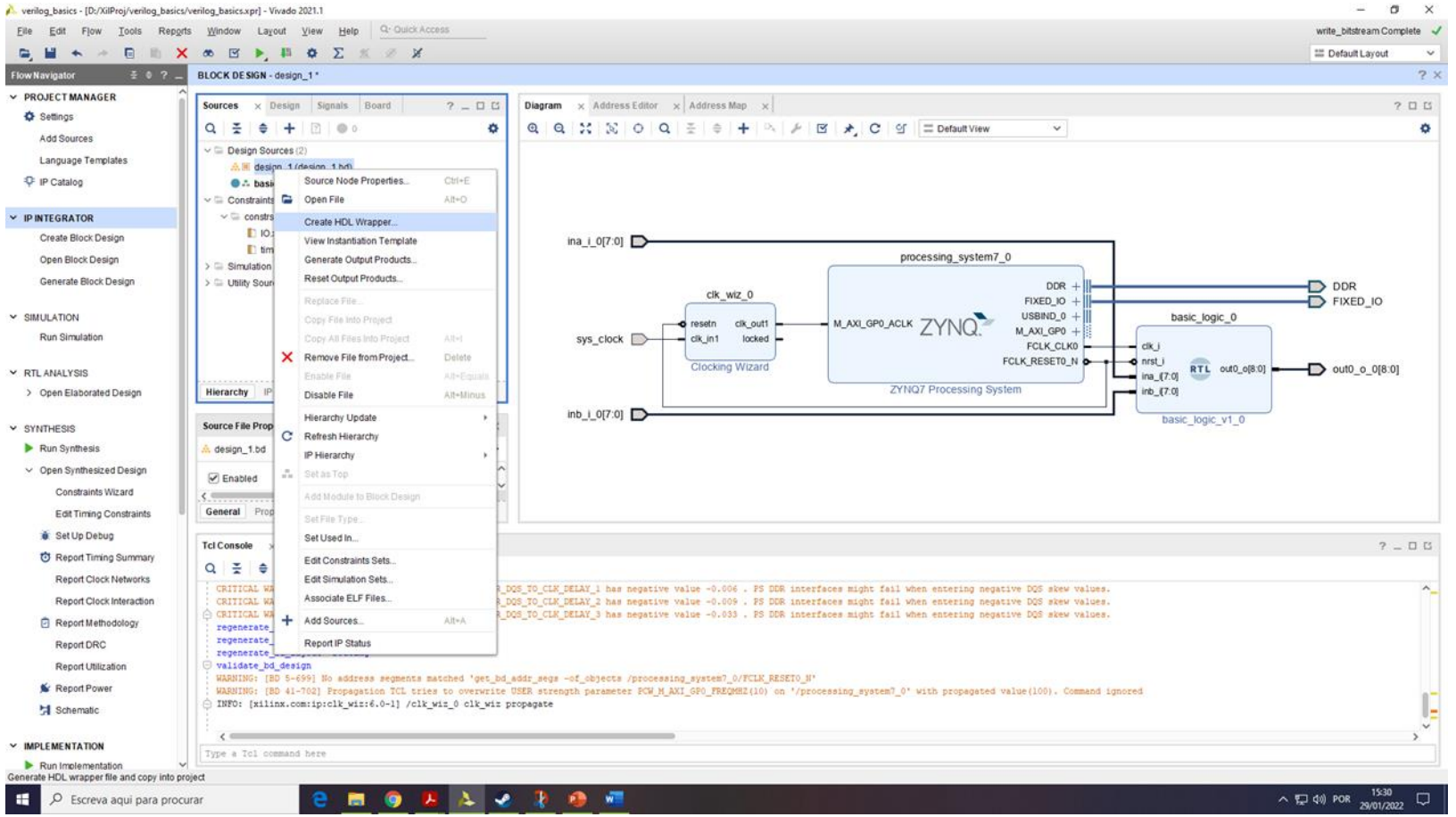
Design Flow

- Finally click on the “Validate Design” icon to make sure everything is ok with the design we have just created. It should validate successfully.



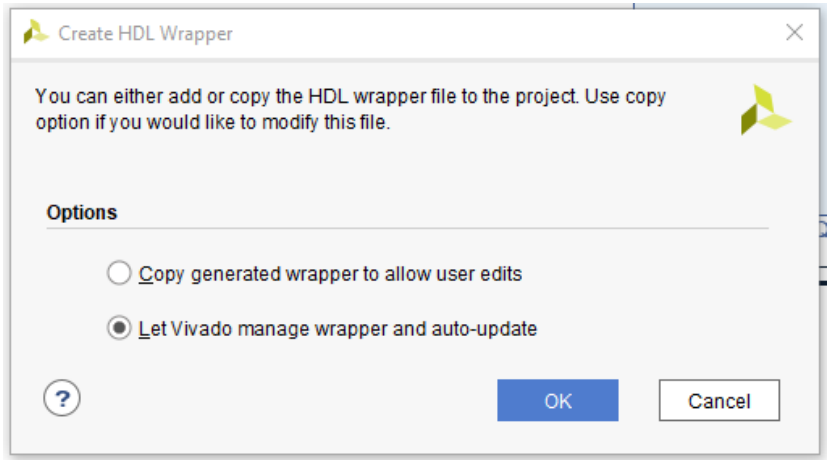
Design Flow

- To finish the Block Design, right-click on the design and select “Create HDL Wrapper…”



Design Flow

- Let Vivado manage the auto-updates and click “OK”
- Congratulations! You have your first block design.
- Try follow the traditional design flow (Synthesis->Implementation->Generate Bitstream) to generate a bitstream and program the FPGA and configure the Arm core
- Are there any errors? Why? What must be changed?



Design Flow

- Design Rule Check (DRC)
  - Verifies if the design meets the constraints imposed for a given technology
  - It guarantees that the design meets the manufacturing requirements (ASIC)
  - Without it, the fabricated design may not function properly or even fail during manufacturing
  - Given by the Vendor or Fabrication facility
  - Among others, some DRC rules include:
  - Minimum Width
  - Minimum Spacing

Design Flow

- Design Rule Check (DRC)
  - Minimum Area
  - Wide metal jog
  - Misaligned via wire
  - Special notch spacing
  - End of line spacing
  - In FPGA DRC checks if port mappings are according to specifications along with other physical verifications regarding routing, cell placement and resources selection (for example a clock signal that was not mapped to a clock capable pin)

Design Flow

- Layout Vs Schematic (LVS)
  - Compares the extracted netlist from the layout with the original schematic netlist
  - The schematic is a drawing with information related to the existing connections
  - The layout has the exact physical location of every module/cell and how they are connected using physical wires
  - Both should match to guarantee the circuit to be fabricated is indeed the one that was designed (some errors may be introduced by the tools or the design engineer when optimizing the design)

THANK YOU!

ANY QUESTIONS?

rui.machado@dtx-colab.pt