MODULE 6

SoC Design

Contents

- Introduction to hardware – software codesign
- Introduction to
  - Qsys and Intel Quartus prime tool
  - Nios II Software Build Tools for Eclipse
- Incorporate custom peripherals & instructions into an embedded system.

Hardware – software codesign

- An approach in computing and engineering that involves designing both hardware and software components of a system simultaneously, in a collaborative manner.
- In other words, “Meeting System level objectives by exploiting the synergism of hardware and software through their concurrent design”
- This method is used to optimize
  - system performance,
  - improve efficiency, and
  - ensure that the hardware and software work seamlessly together.

Microelectronics trends

- Better device technology
  - reduced in device sizes
  - more on chip devices > higher density
  - higher performances
- Higher degree of integration
  - increased device reliability
  - inclusion of complex designs

Digital Systems

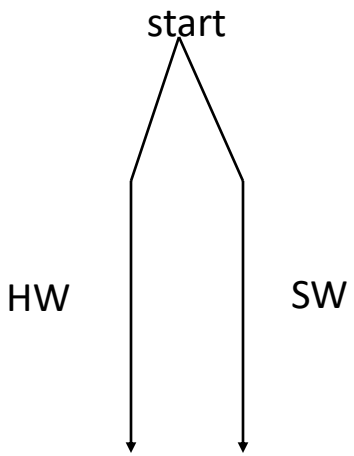
Judged by its objectives in application domain

- Performance
- Design and Manufacturing cost
- Ease of Programmability

It depends on both the hardware and software components

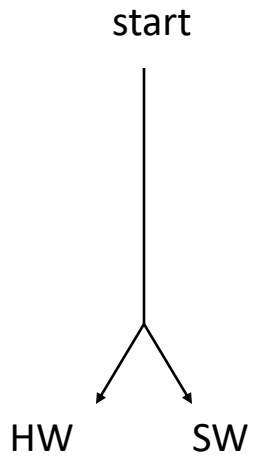
Concurrent design

Traditional design flow



Designed by independent groups of experts

Concurrent (codesign) flow



Designed by Same group of experts with cooperation

Motivation

Trend toward smaller mask-level geometrics leads to:

- Higher integration and cost of fabrication.
- Amortize hardware design over large volume productions.

  Suggestion:

  Use software as a means of differentiating products based on the same hardware platform.

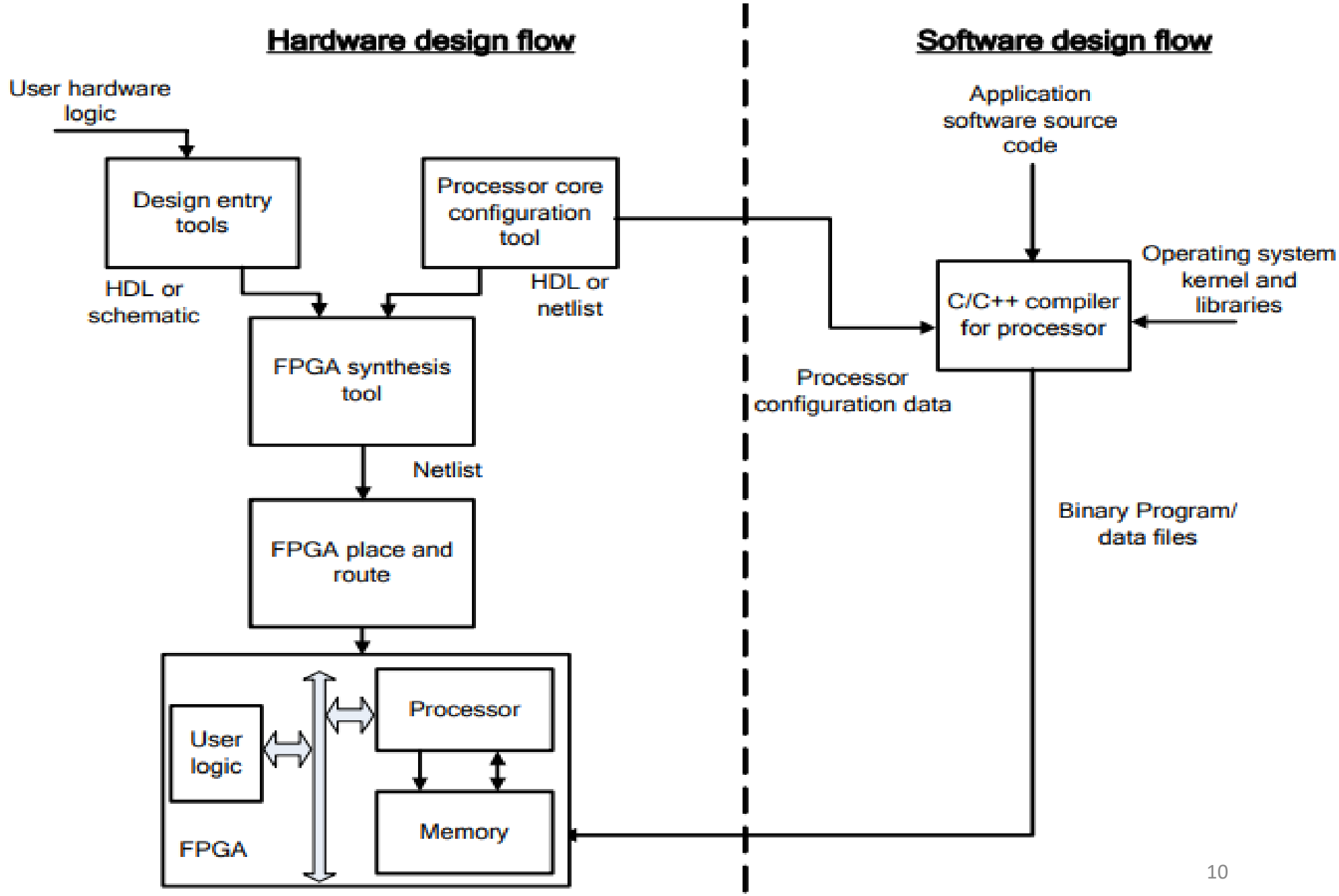
Why co-design?

- Performance Optimization: By considering both hardware and software requirements at the same time, designers can tailor solutions that make the best use of the resources available.
- Energy Efficiency: Especially in embedded systems (like mobile phones or IoT devices), codesign helps balance power consumption between hardware and software.
- Flexibility: Some functions might be better suited for software implementation, while others are more efficient in hardware. Codesign allows shifting between hardware and software as needed.
- Time and Cost Savings: By developing both aspects in tandem, potential mismatches are caught early, saving time and costs in the development process.

Codesign Process

- Specification:
  - The system’s requirements are clearly defined, focusing on both software functionality and hardware capabilities.
- Partitioning:
  - The system’s functions are divided into tasks handled by either hardware or software. Decisions on what should be hardware or software are made based on performance, power, and cost considerations.
- Simulation and Verification:
  - Both hardware and software components are simulated to ensure they perform as expected and interact seamlessly.
- Implementation:
  - The hardware is built (often using tools like VHDL or Verilog for hardware description), while software is developed in languages like C or Python. The components are then integrated.

Hardware–software design flow



IP cores

- Predesigned, pre verified silicon circuit block, usually containing 5000 gates, that can be used in building larger application on a semiconductor chip.
- Complex macro cells implementing instruction set processors (ISP) are available as cores:
  - Hardware (core)
  - Software (micro kernels)

IP core reuse

- Cores are standardized for reuse as system building blocks.
- Rationale: leveraging the existing software layers including OS and applications in ES.
- Results:
  1. **Customized VLSI chip with better area/ performance/ power trade-offs**
  2. **Systems on Silicon**

FPGAs

- FPGA circuits can be configured on-the-fly to implement a specific software function with better performance than on microprocessor.
- FPGA can be reprogrammed to perform another specific function without changing the underlying hardware.
- *This flexibility opens new applications of digital circuits.*

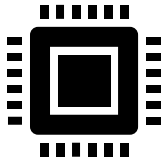
Applications

- General purpose computing system
    - usually self contained and with peripherals
    - Information processing systems
- Dedicated control system
    - part of the whole system, Ex: digital controller in a manufacturing plant
    - also, known as embedded systems

Codesign Tools

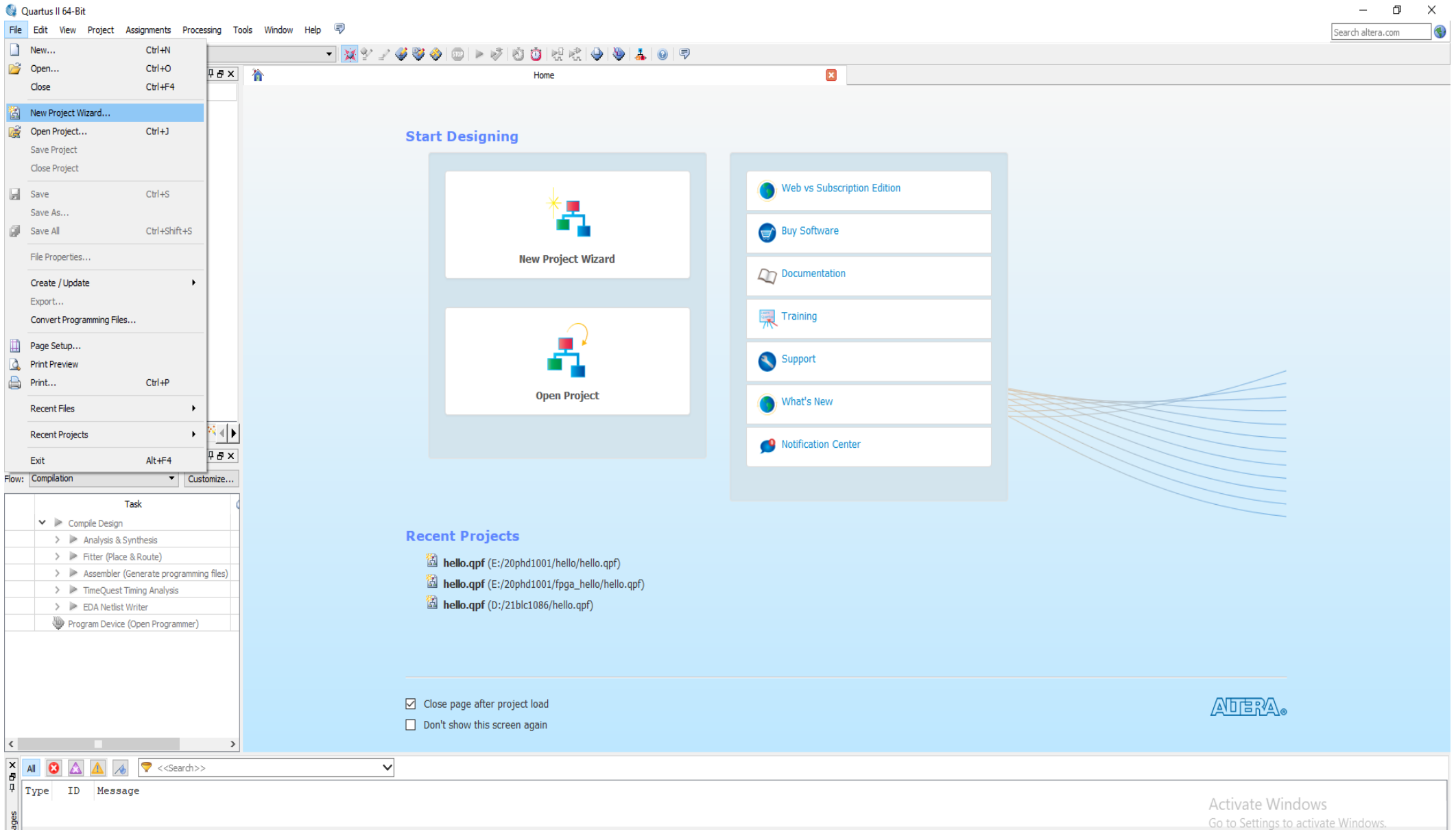
- MATLAB/Simulink:
  - For simulating and modeling system behaviour.
- SystemC:
  - A language for system-level modeling.
- Cadence, Synopsys, Mentor Graphics:
  - Electronic Design Automation (EDA) tools that provide solutions for codesign, including verification and simulation environments.

Incorporate custom peripherals & instructions into an embedded system.

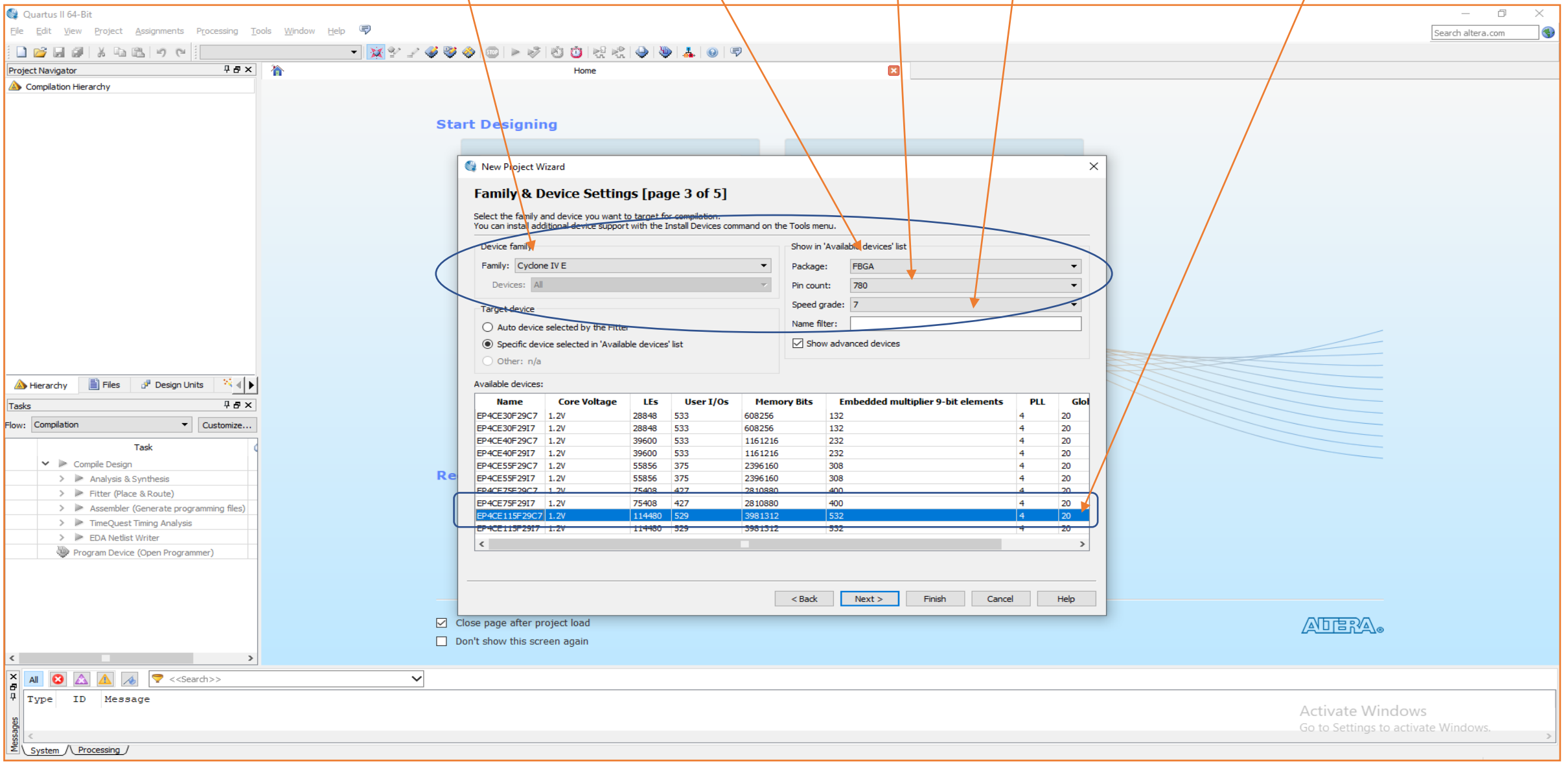


1. Open “Quartus Prime Tool”. Create a “New Project Wizard”.

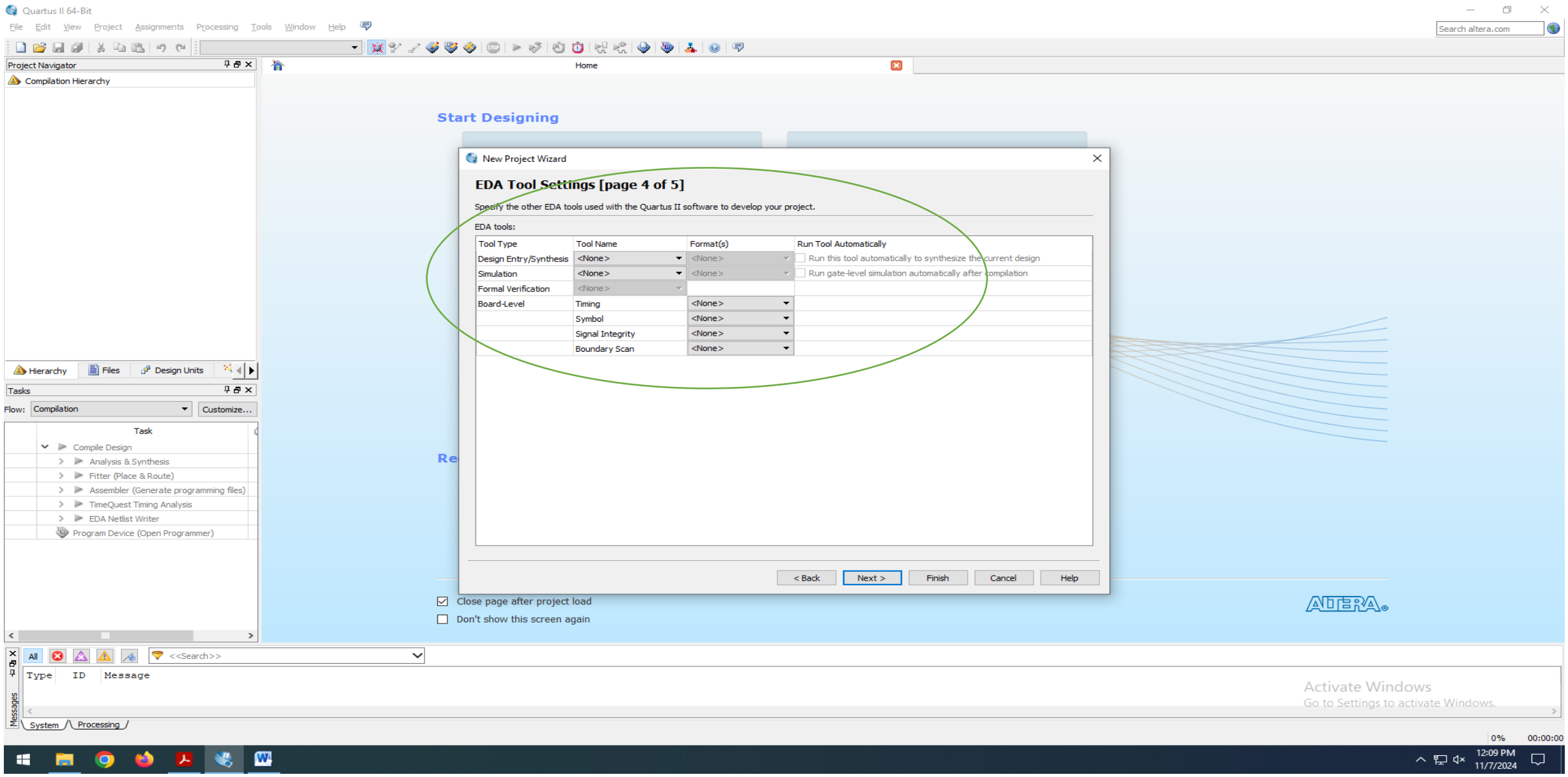
File → New Project Wizard



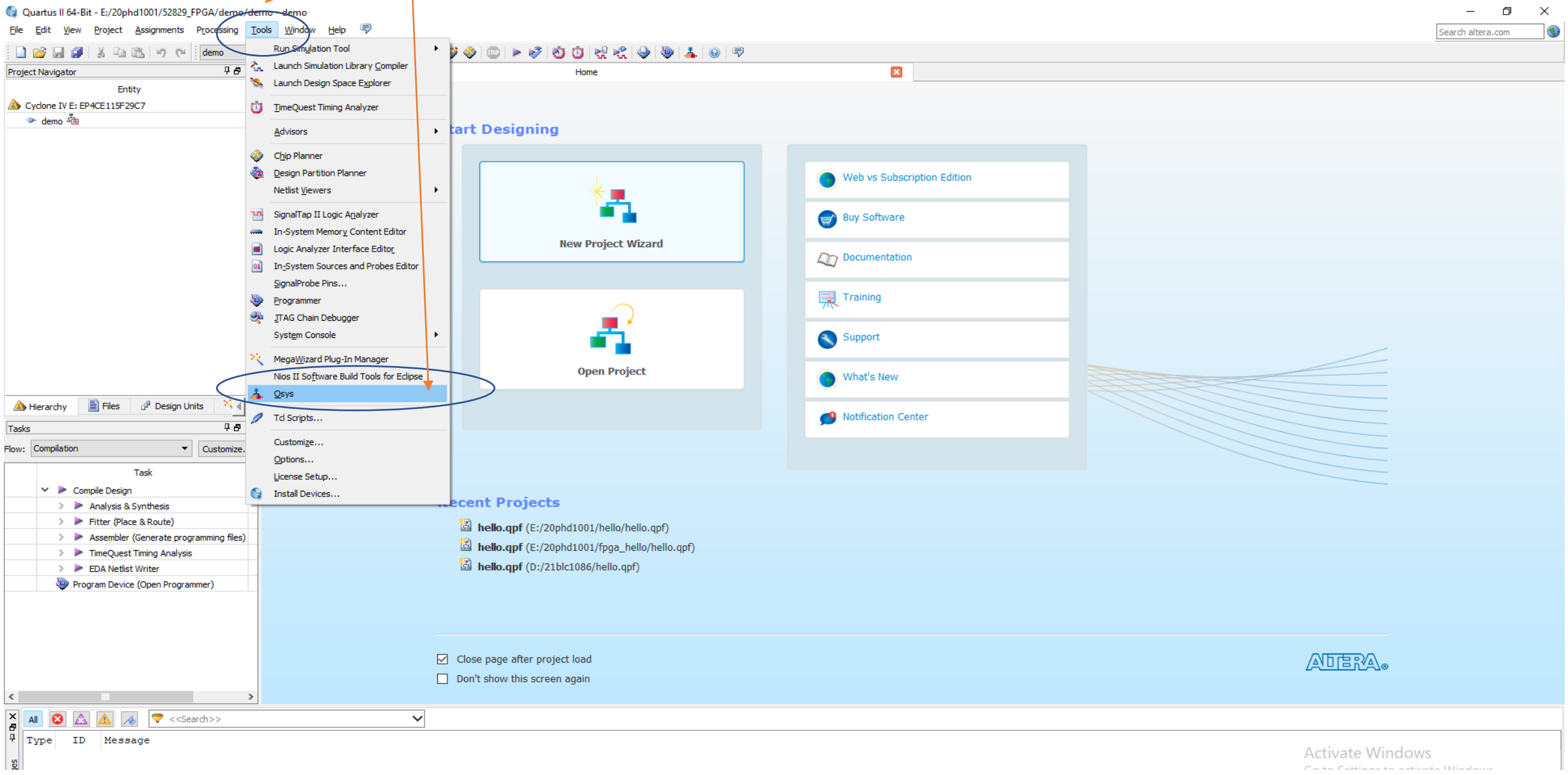
2. Choose the Device Family *Cyclone IV E* , Package *FBGA*, Pin Count *780,* Speed *7* and the Device *EP4CE115F29C7*.



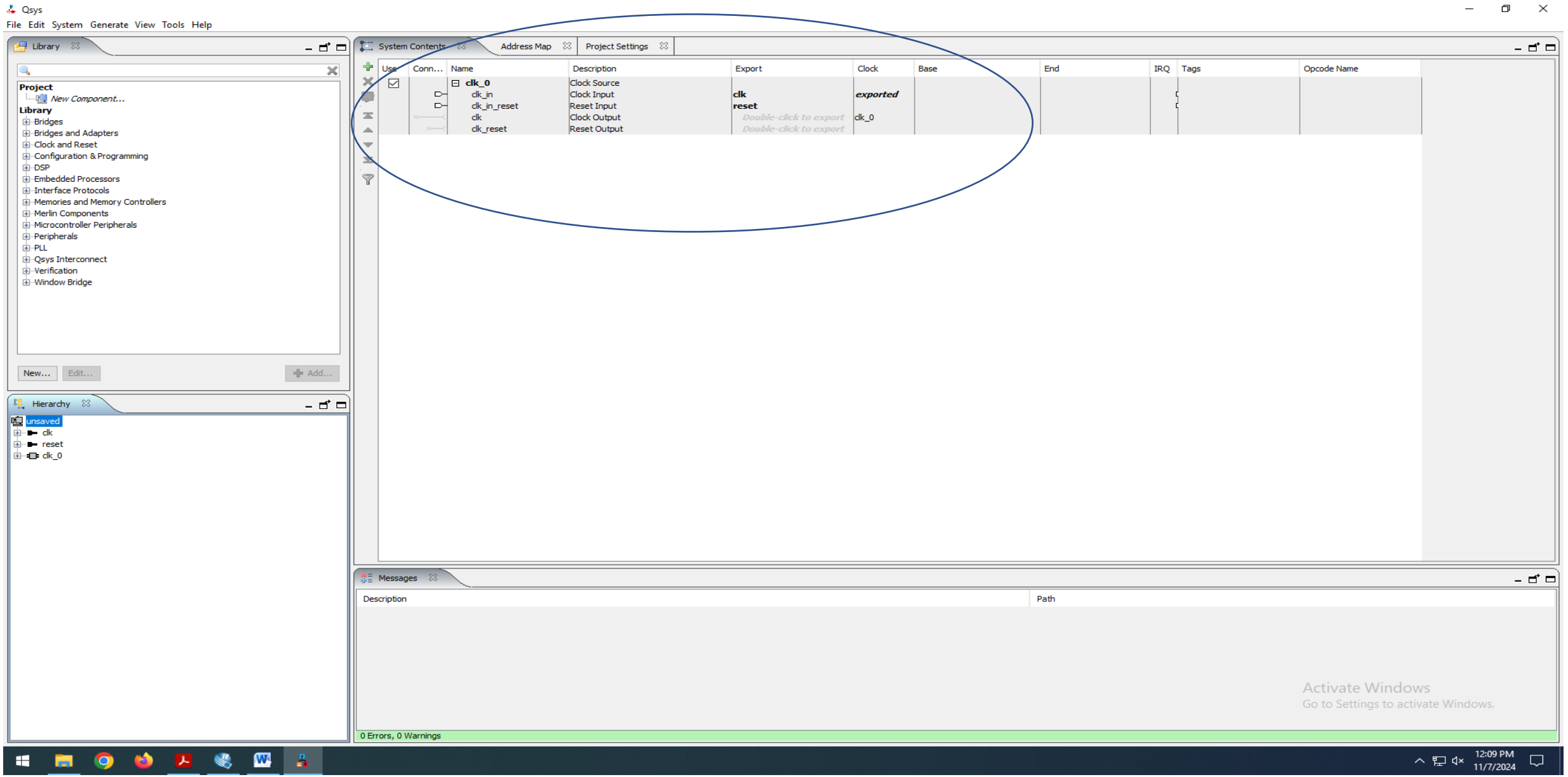
3. Specify all EDA tools as “None”. Finish setup.



4. Open Qsys tool . *Tools → Qsys*



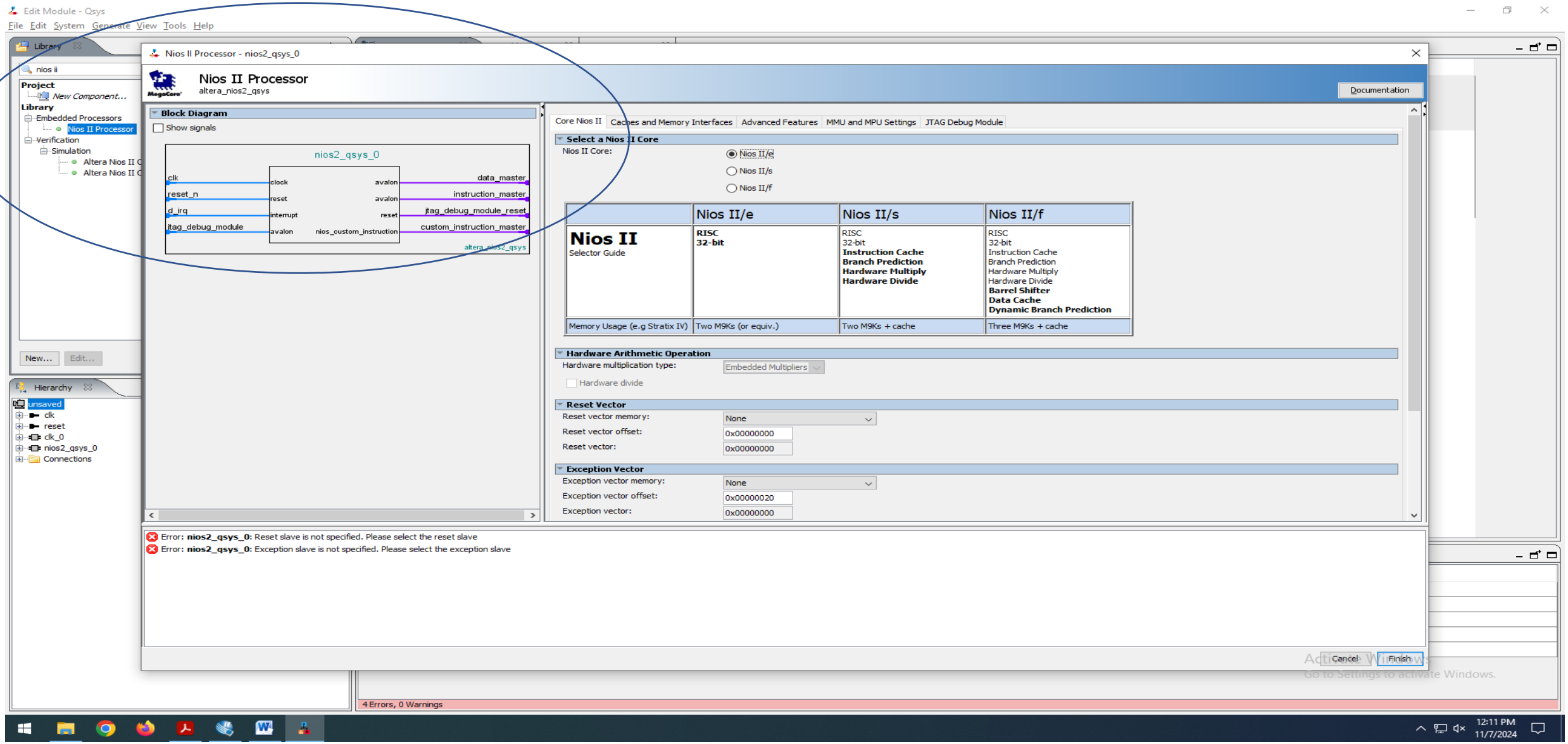
5. Qsys window will show default system contents *(“clk_0”)*



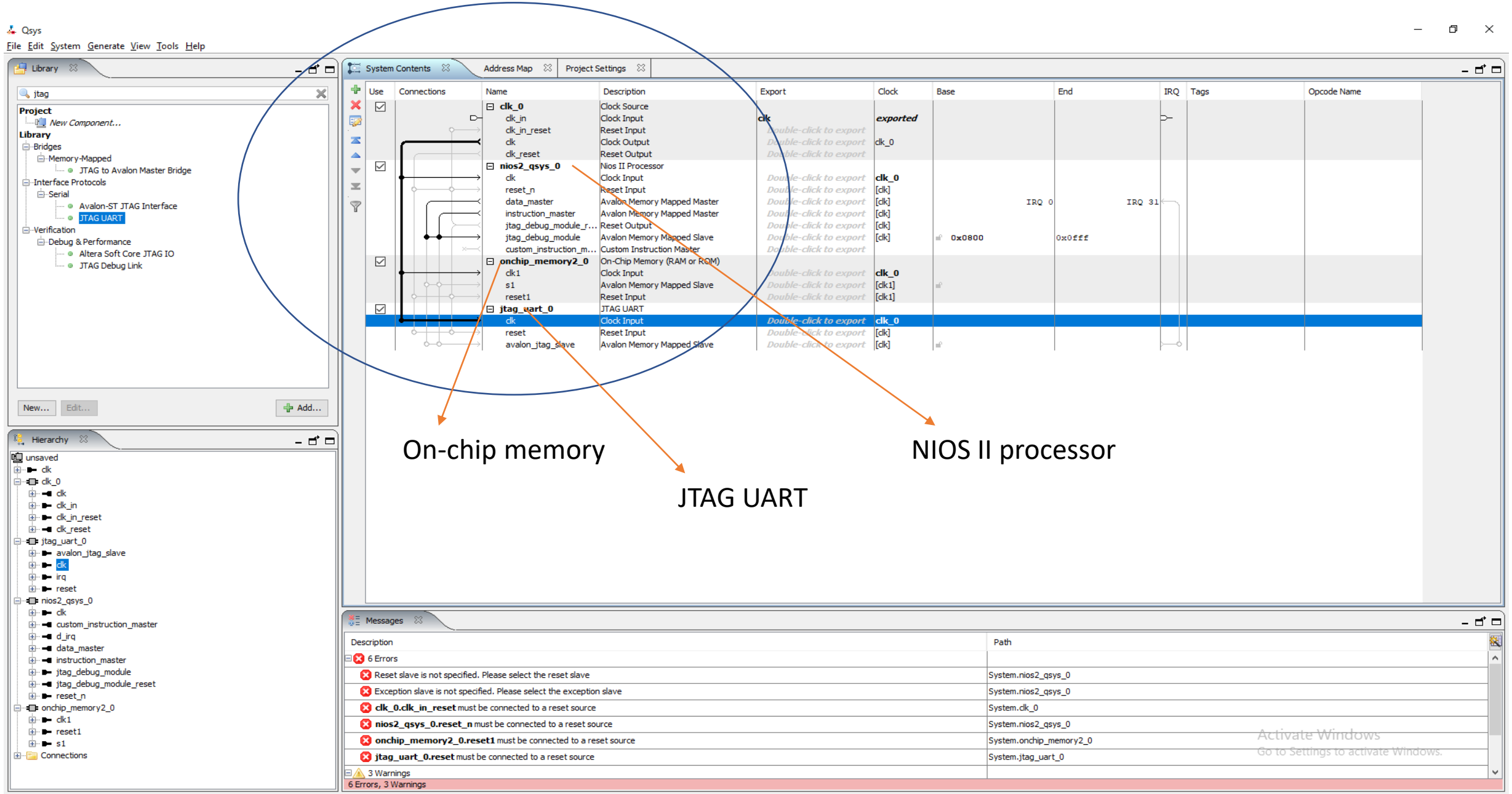
6. Now we build a system that will dispay “Hello”. To build the system we need to add

- NIOS II processor, JTAG for communication, On-chip RAM/ROM

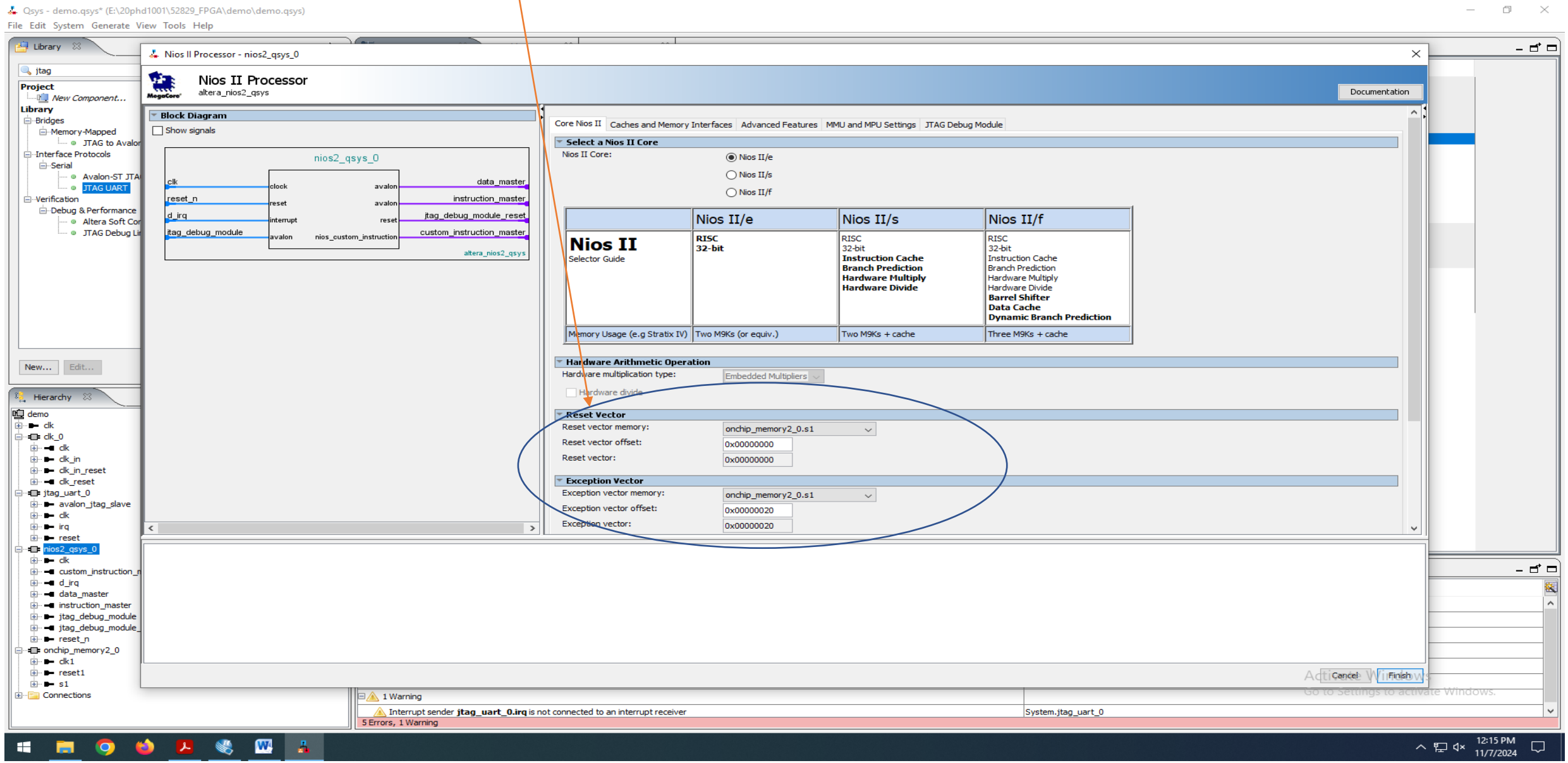
7. *Library → add the required blocks*



8. Once all the blocks are added the window will look like this. Make the necessary connections by clicking on the dots.



9. Once connections are made, double click on the NIOS II. In the NIOS II window choose “Reser Vector memory” and “Exception vector memory” as “onchip_memory……”

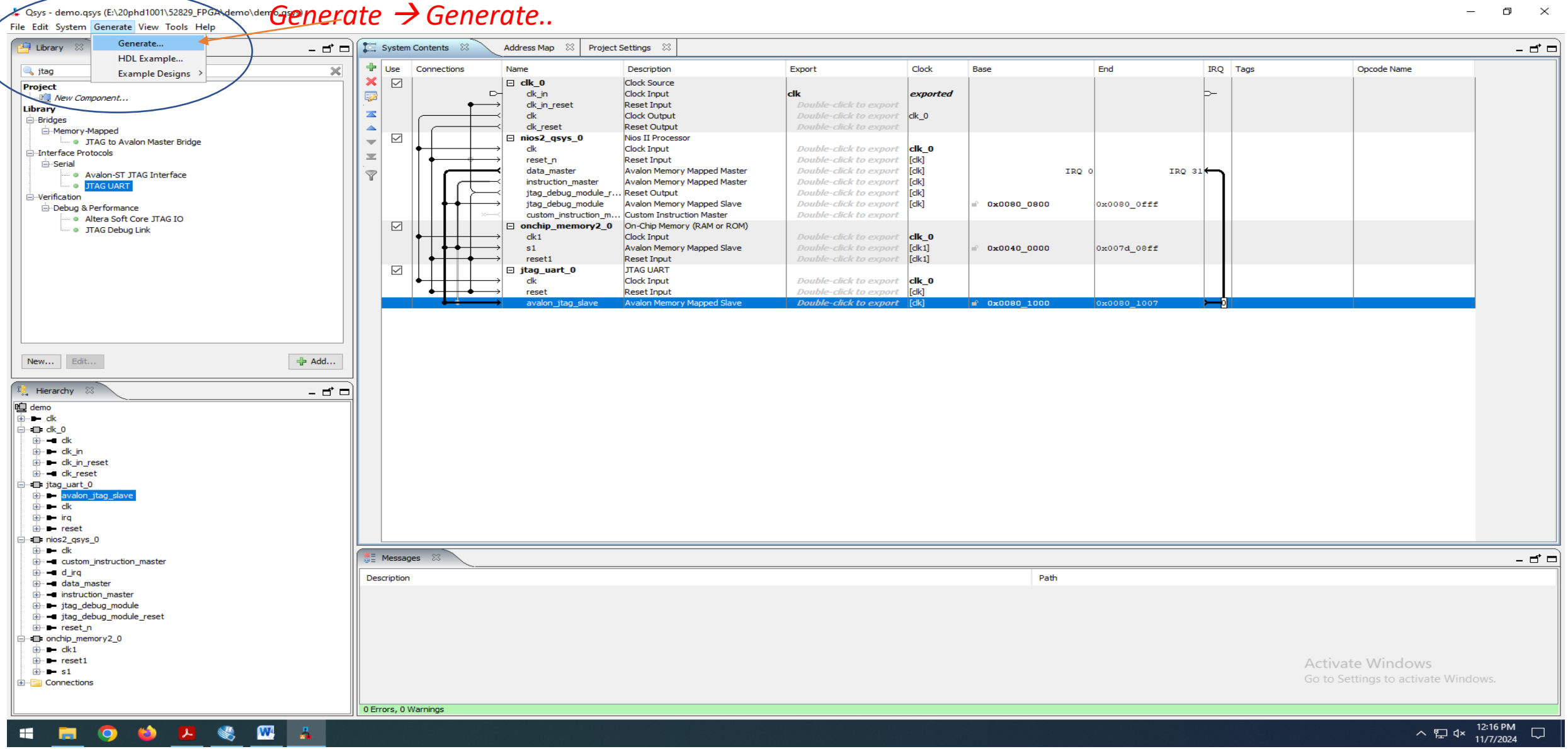


10. Save the design and Generate HDL.

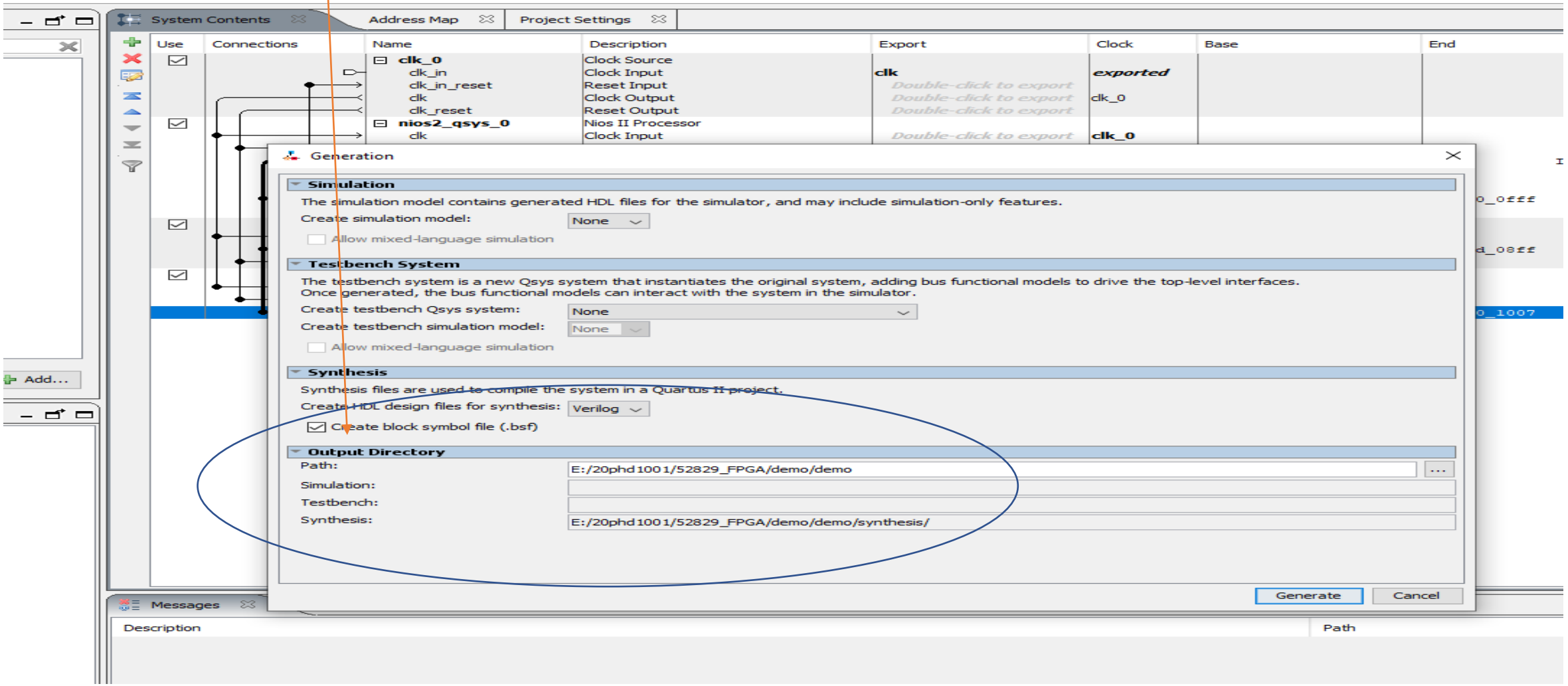
File → Save → <filename.qsys>

System → Assign Base Addresses

Generate → Generate..



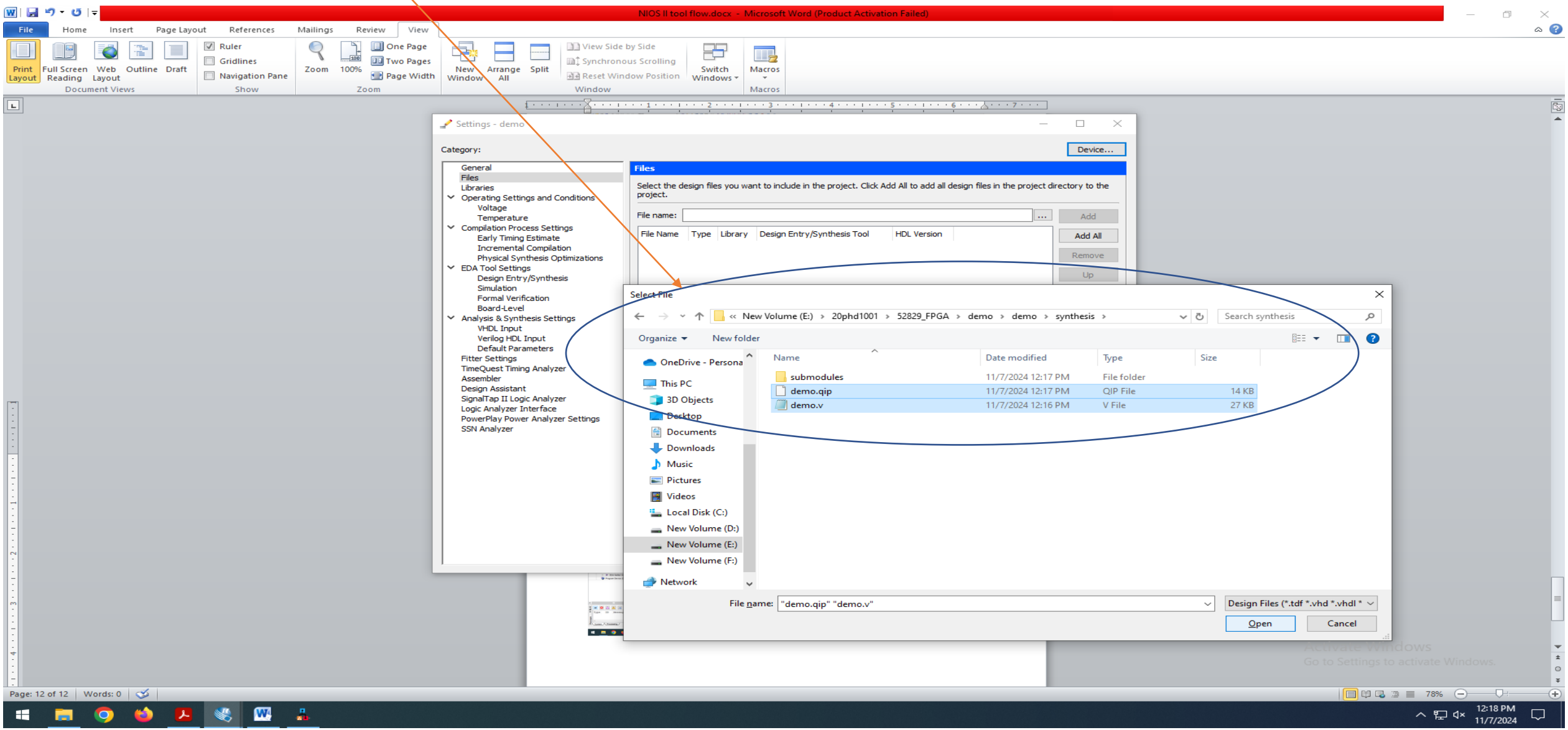
11. Specify the “Output Directory Path” (same as where the project is created). Click “Generate”.



"Designers can also create their own custom peripherals and integrate them into soft processor systems. For performance-critical systems that spend most CPU cycles executing a specific section of code, it is a common technique to create a custom peripheral that implements the same function in hardware. This approach offers a double performance benefit: the hardware implementation is faster than software; and the processor is free to perform other functions in parallel while the custom peripheral operates on data."

Altera on using custom peripherals

12. Now come back to “Quartus “ window. Right click on the Device. *Settings → Files → Browse → Select Files (.v and .ip) …….choose directory……<project name> → synthesis → demo.qip, demo.v*

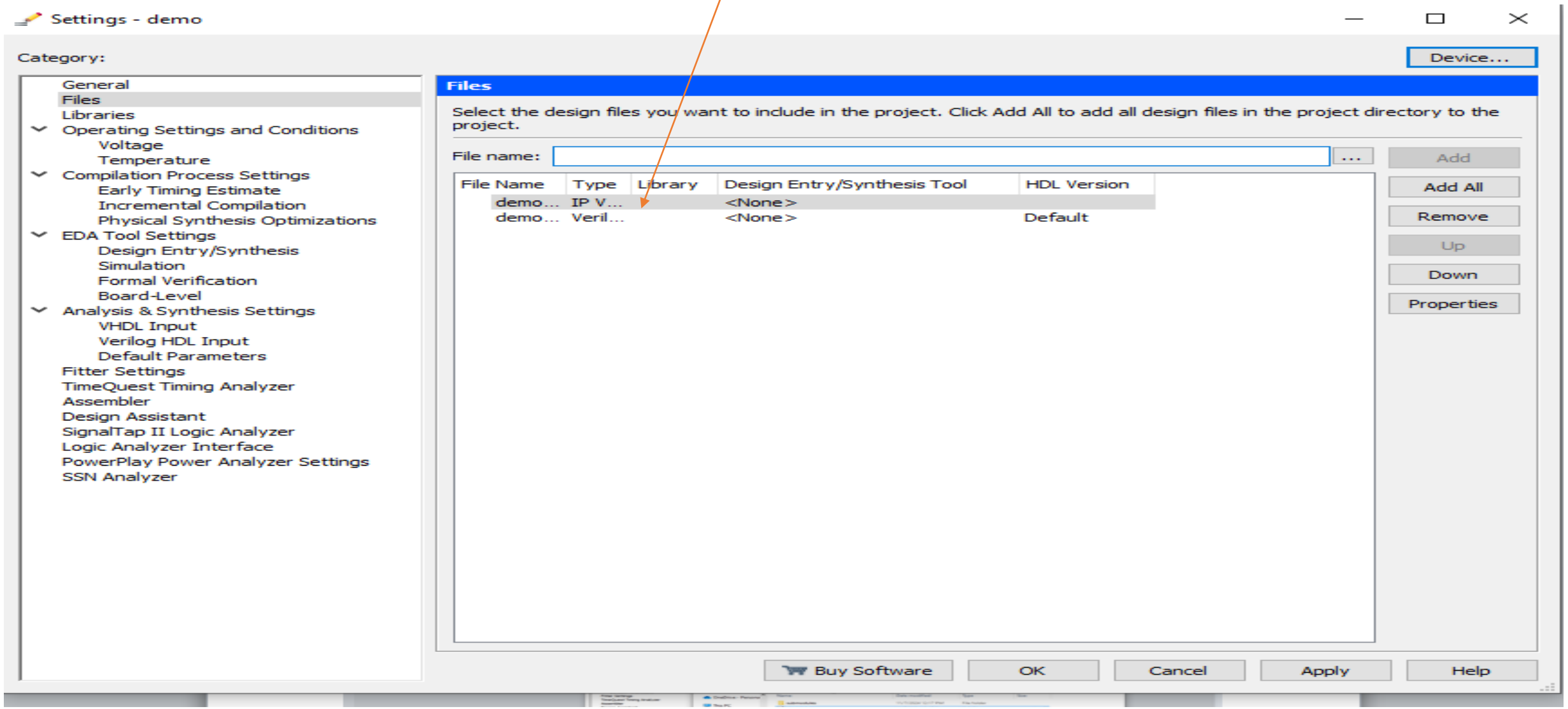


13. Arrange the files.

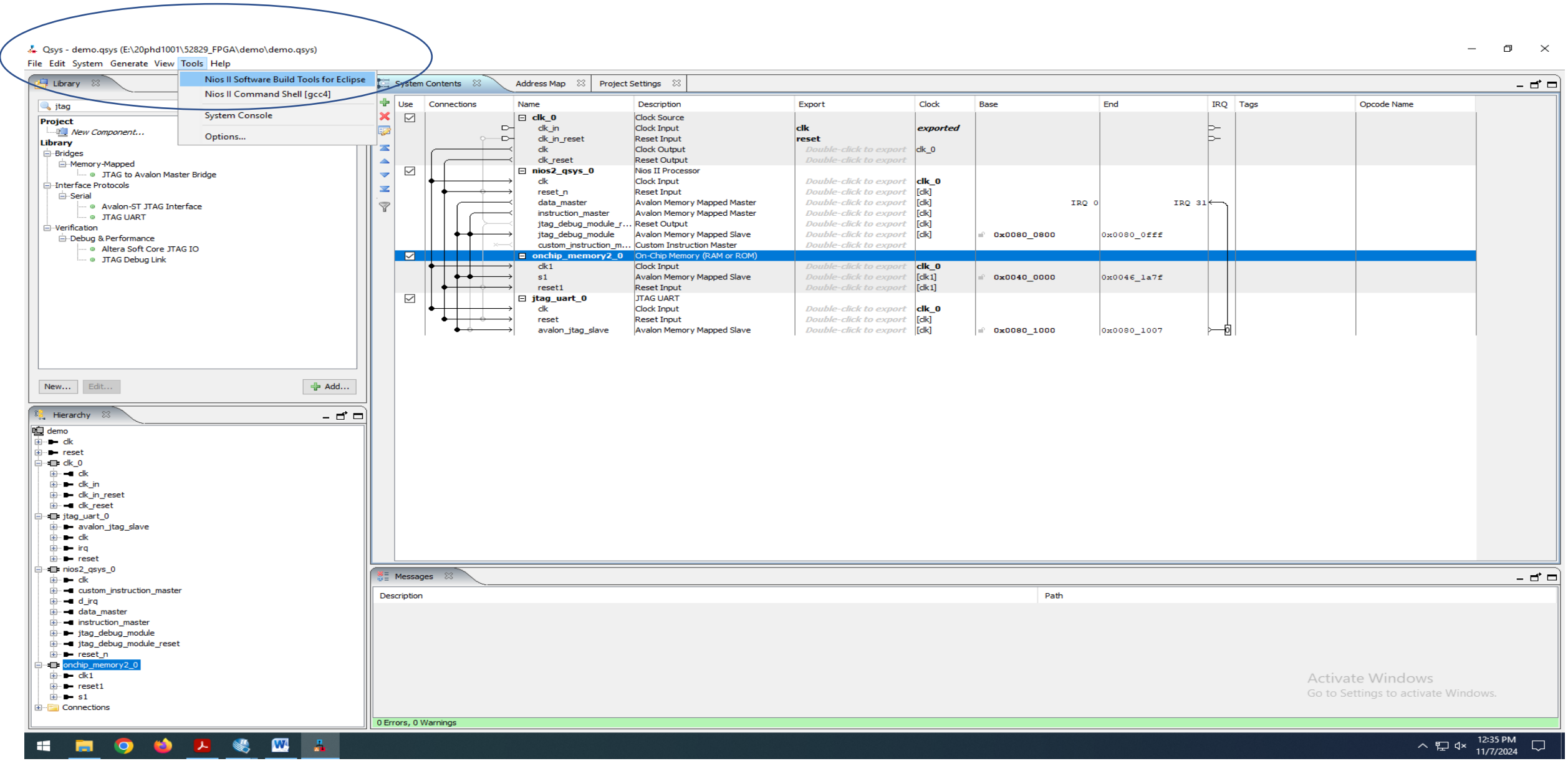
b) Assign pins for clk and reset_n

c) Compile the project.

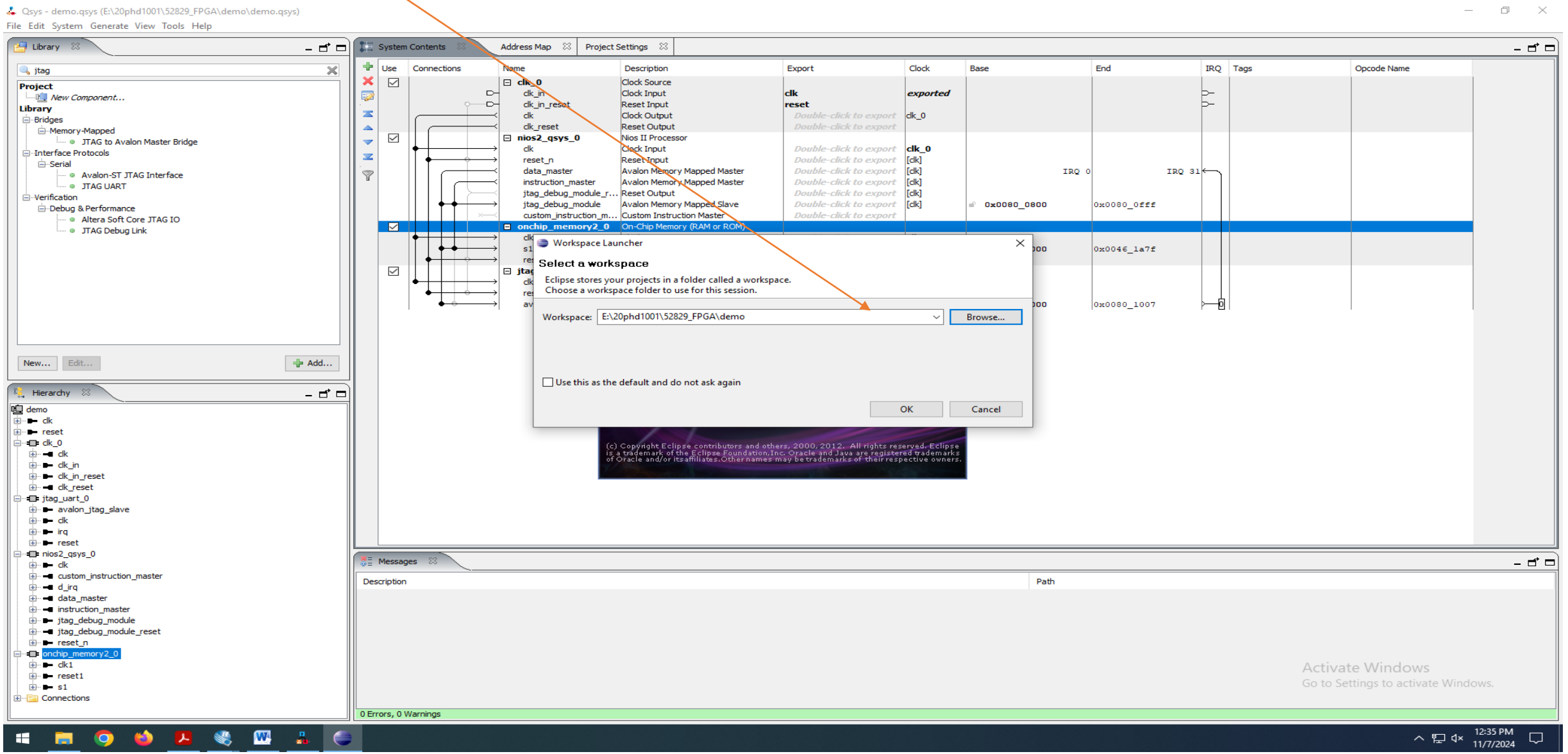
a) Move IP file up and then Verilog file.



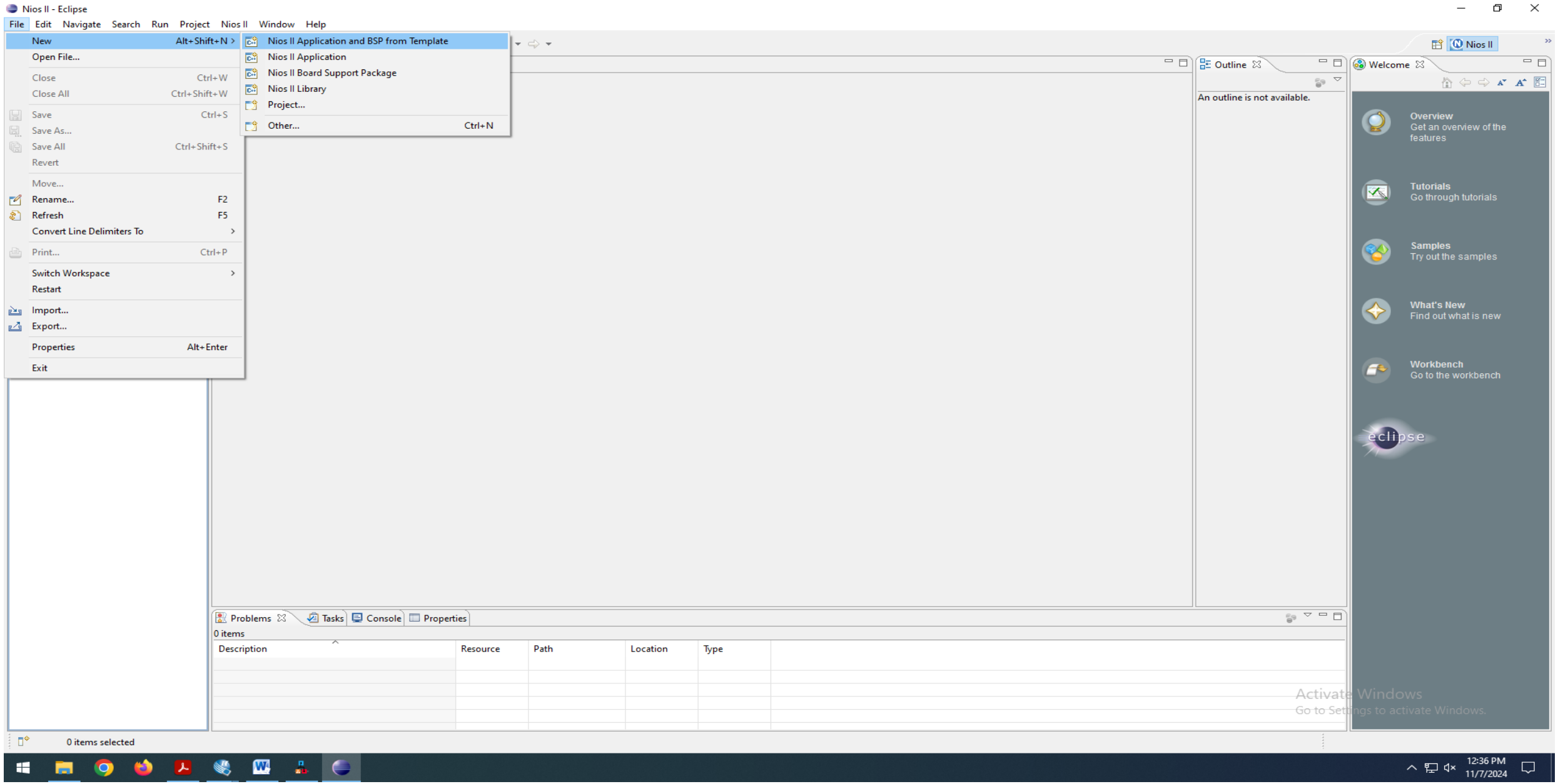
14. Now for the software part, goto *Tools → Nios II Software Build Tools for Eclipse in Qsys window.*



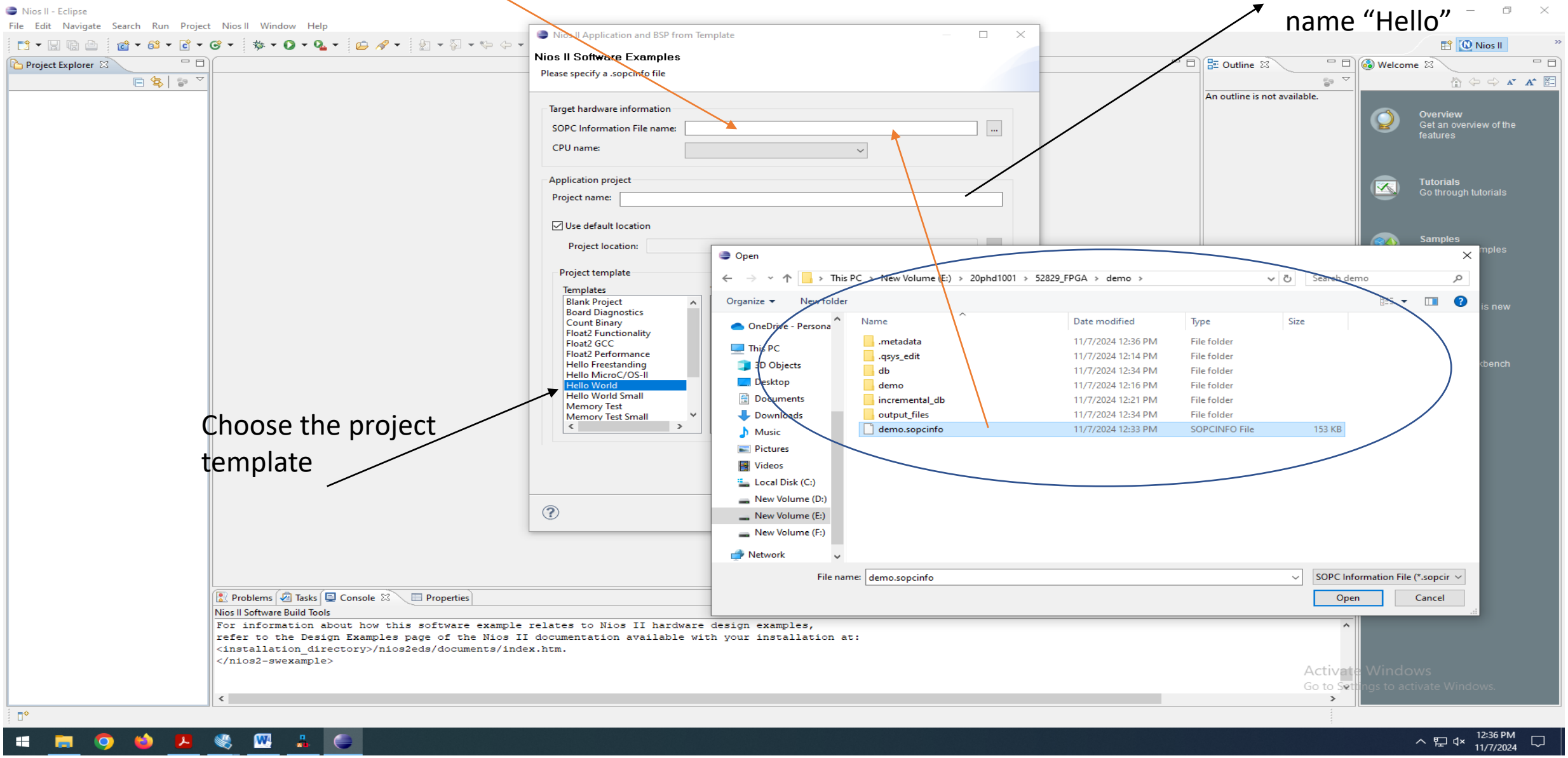
15. In the “workspace launcher” window browse the Workspace to the current working directory where the project was created.



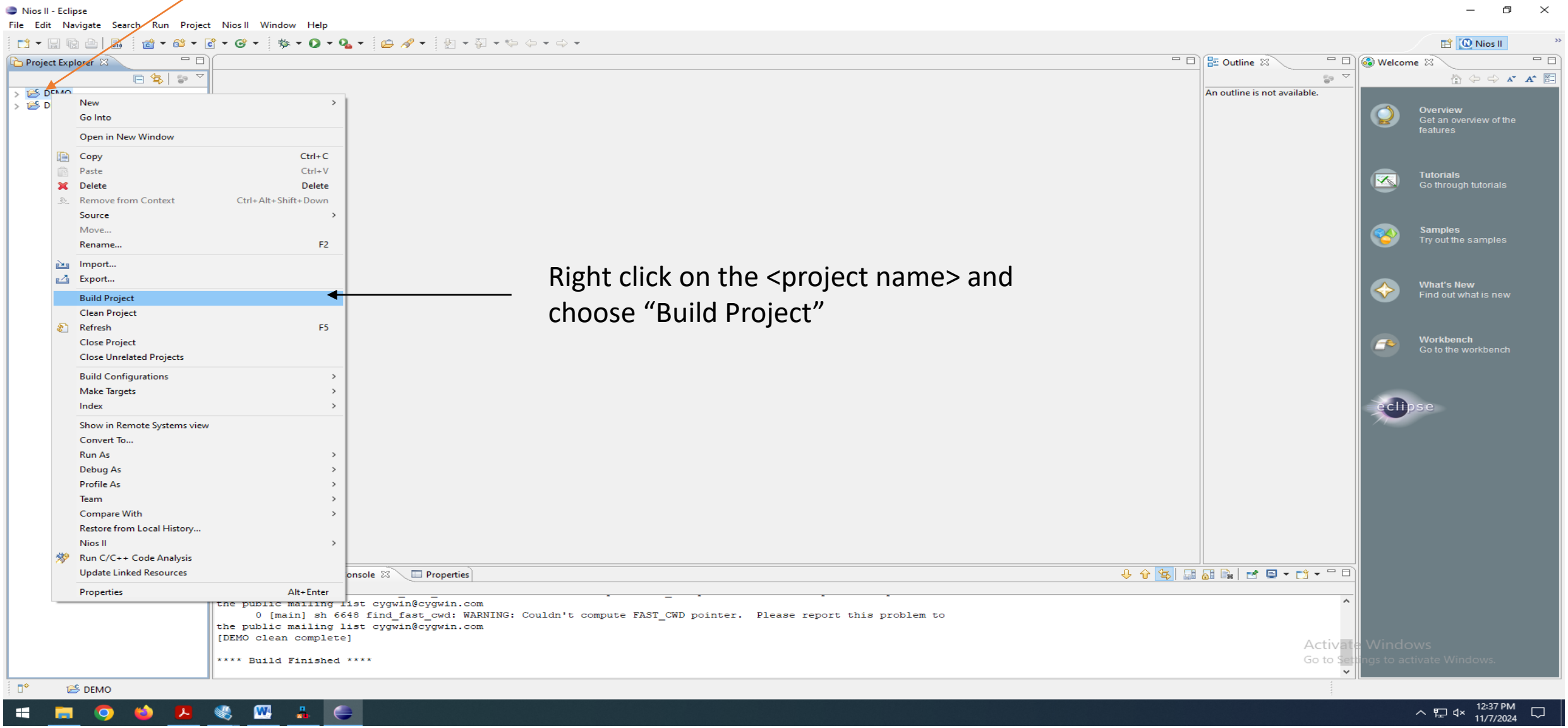
16. Goto *File → New → NIOS II Application and BSP template*



17. Browse and import "*SOPC Information File*" from the working directory which is created after compilation.

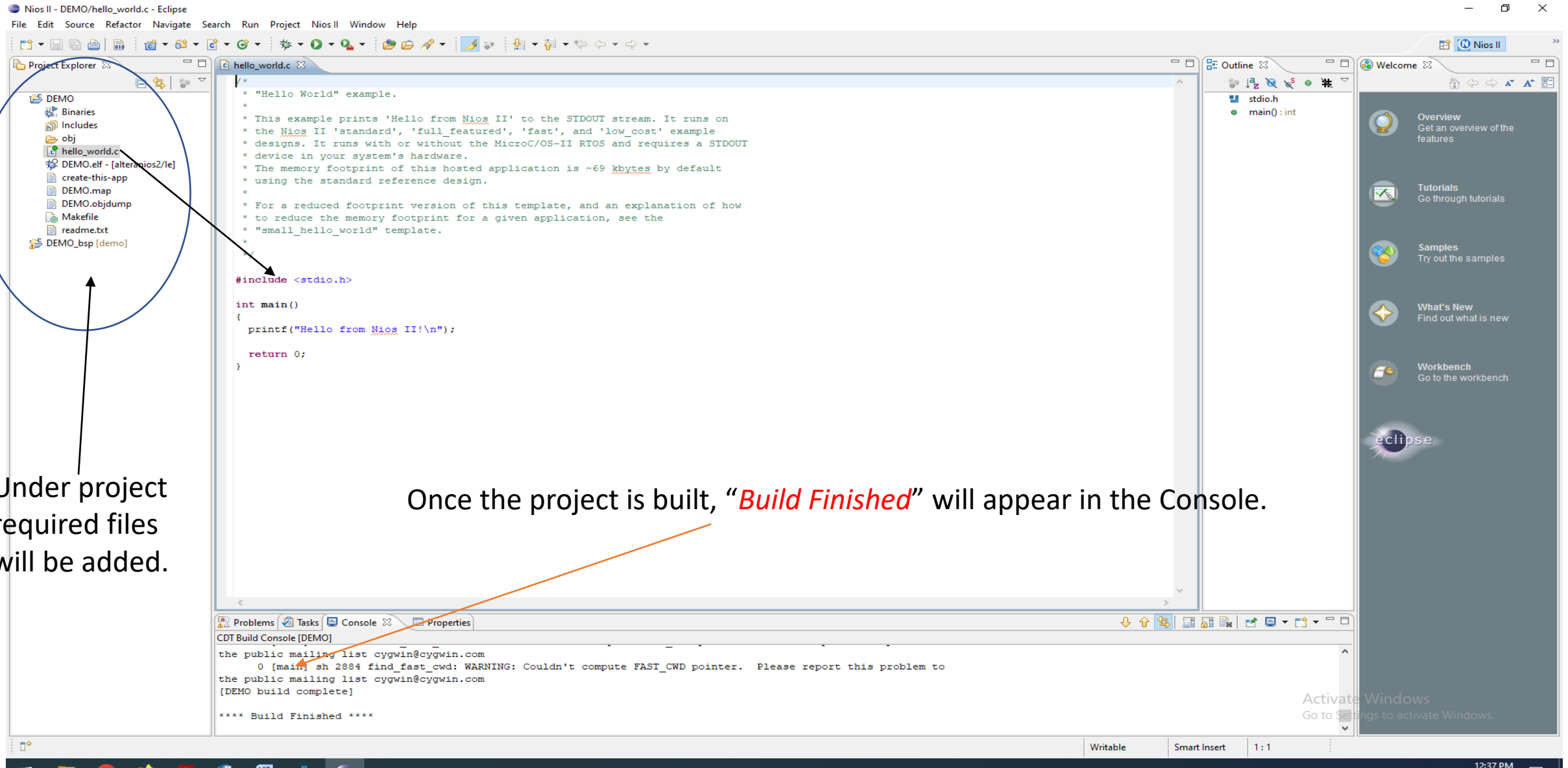


18. Now the project name will appear under “Project Explorer”.



Right click on the <project name> and choose “Build Project”

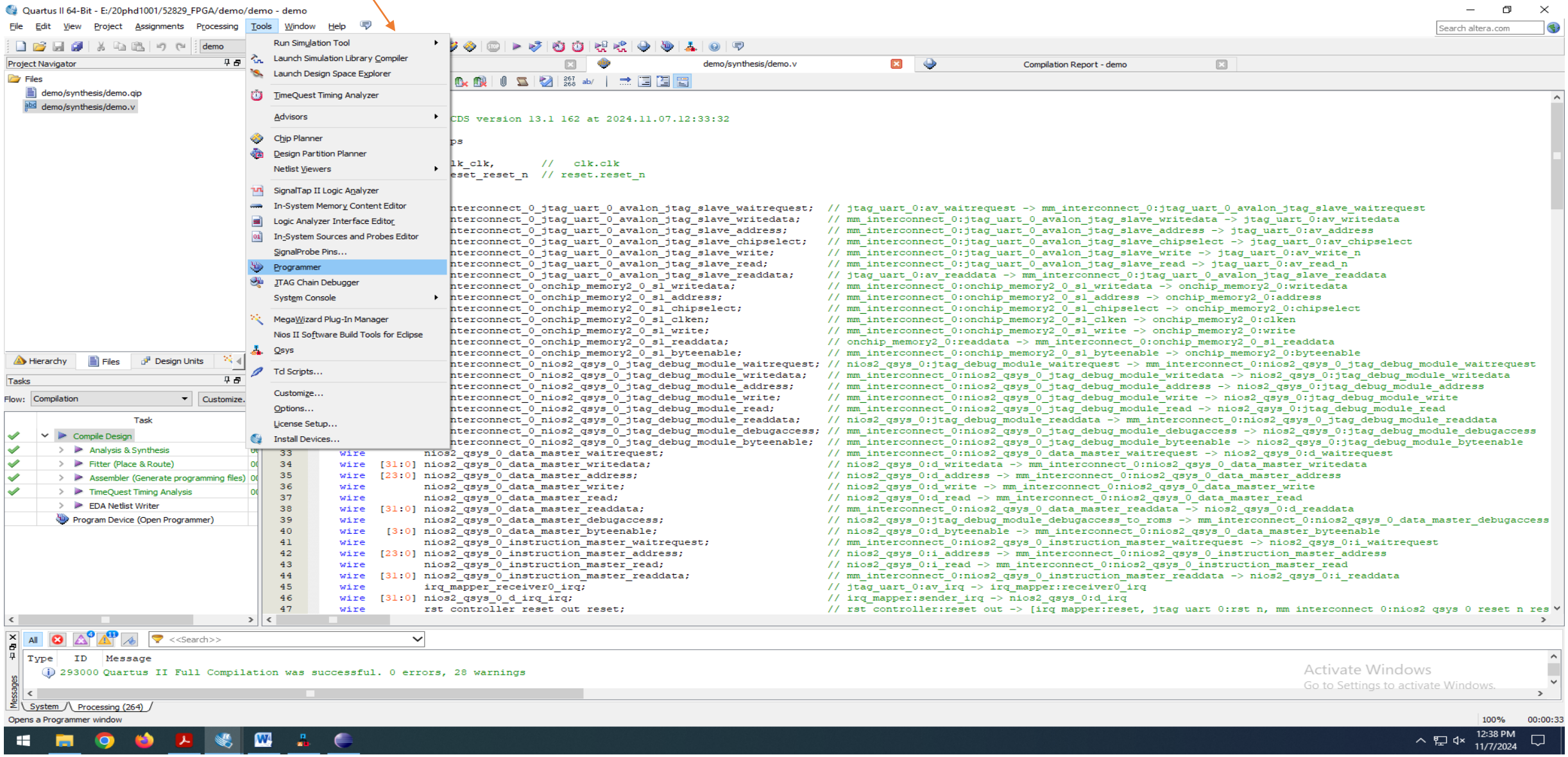
19. Now the project is built.



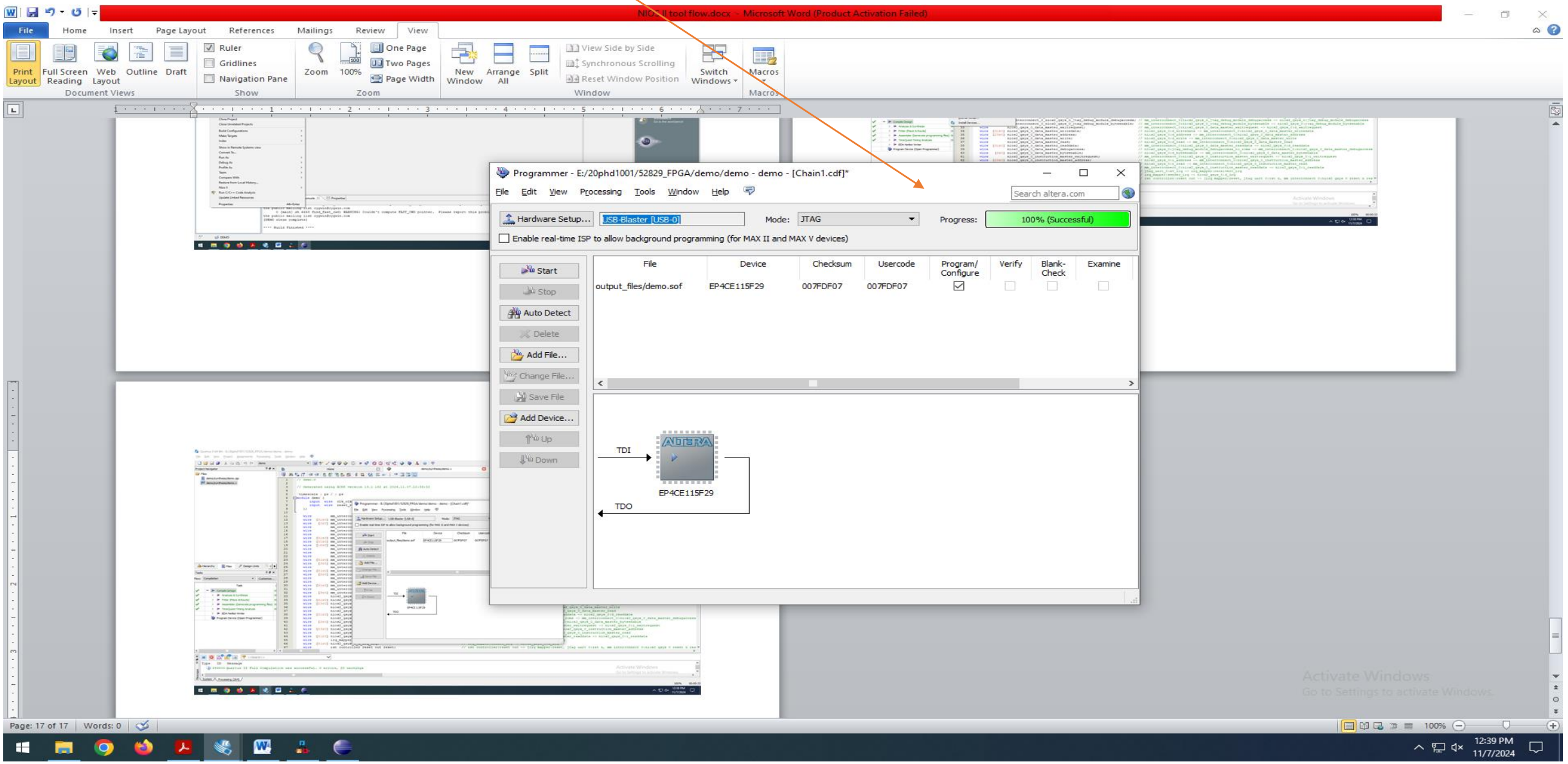
Under project required files will be added.

Once the project is built, "*Build Finished*" will appear in the Console.

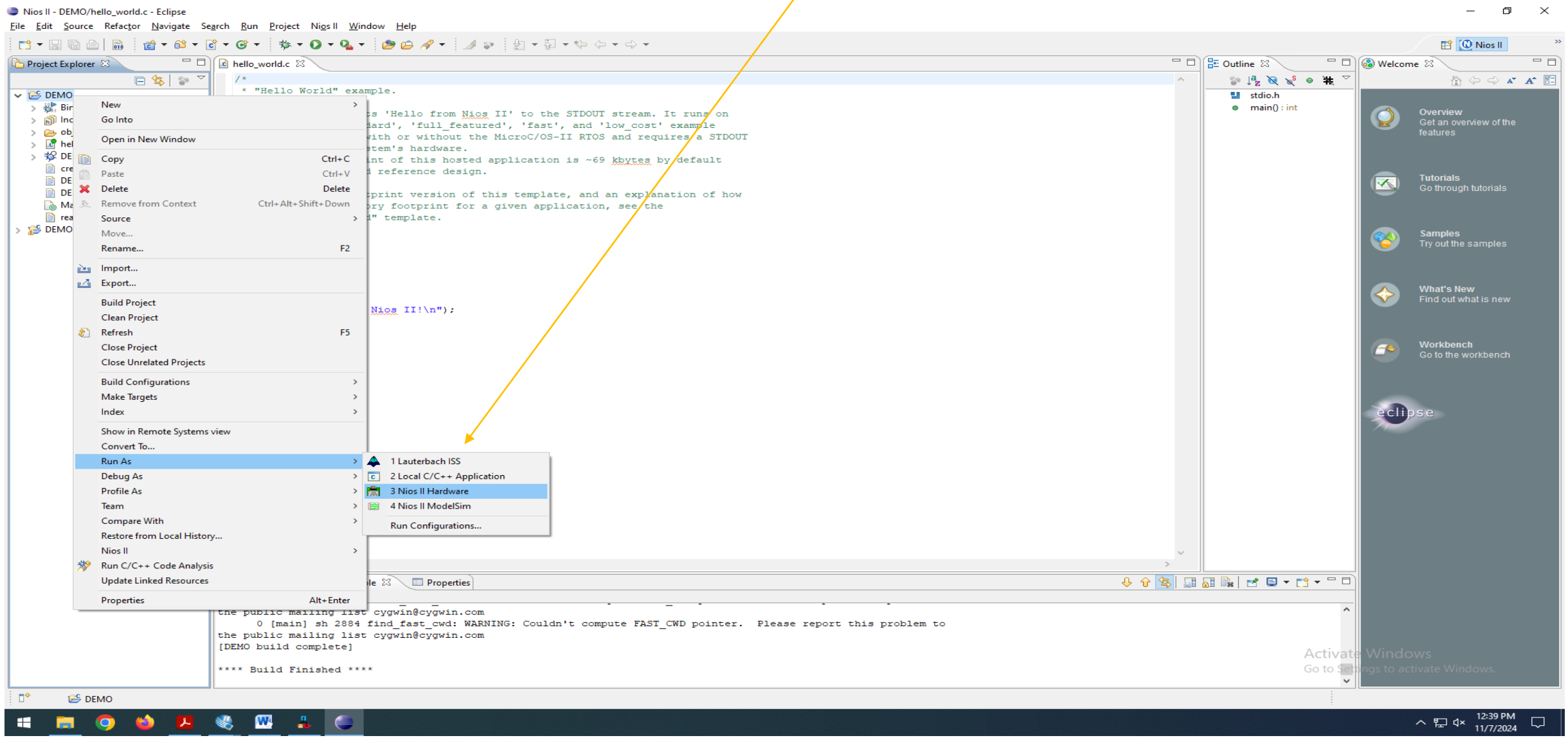
20. Get back to the “Quartus Prime” window, and program the device. Make sure that the Device is connected and switched ON. Tools → Programmer



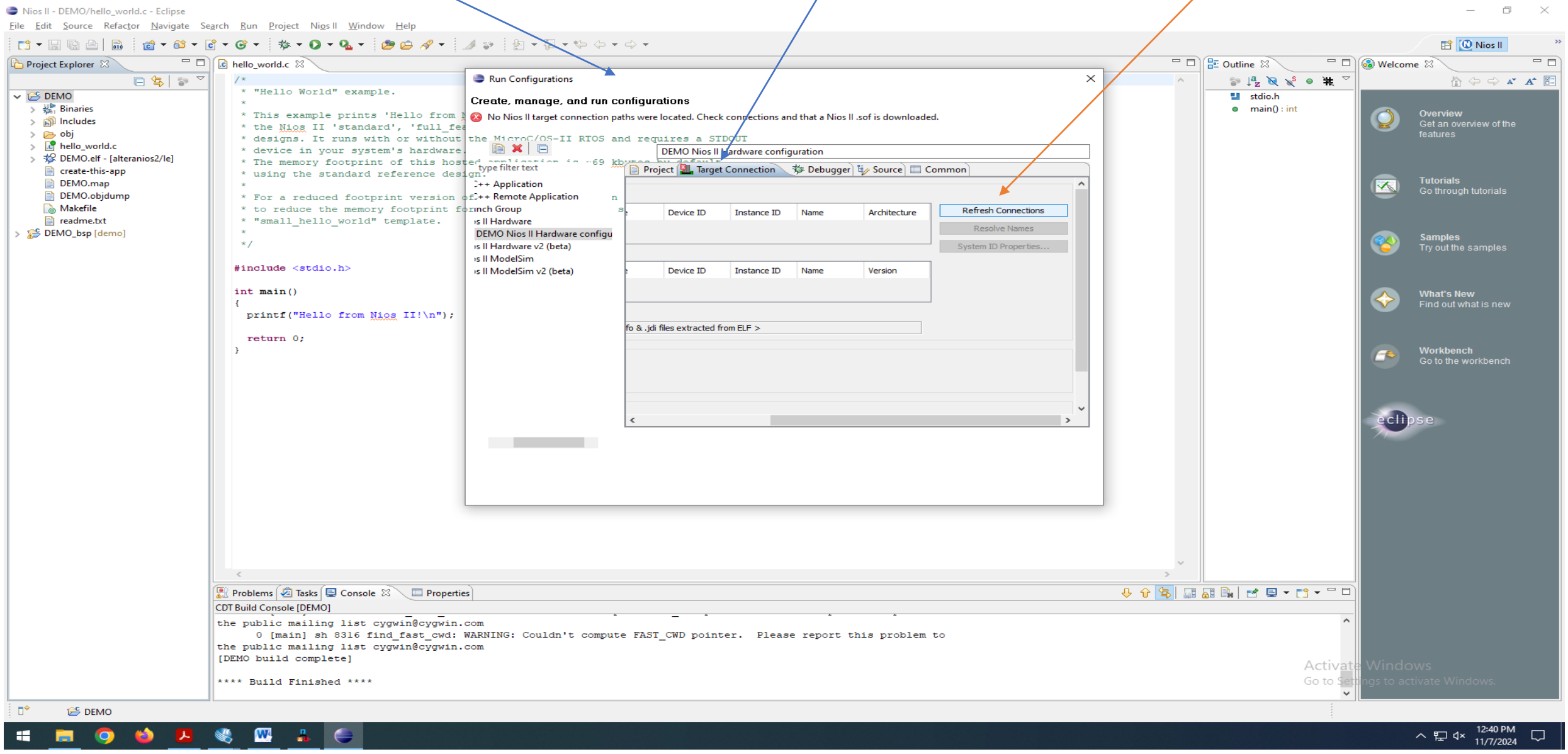
Once Configuration is done. Below window will pop-up.



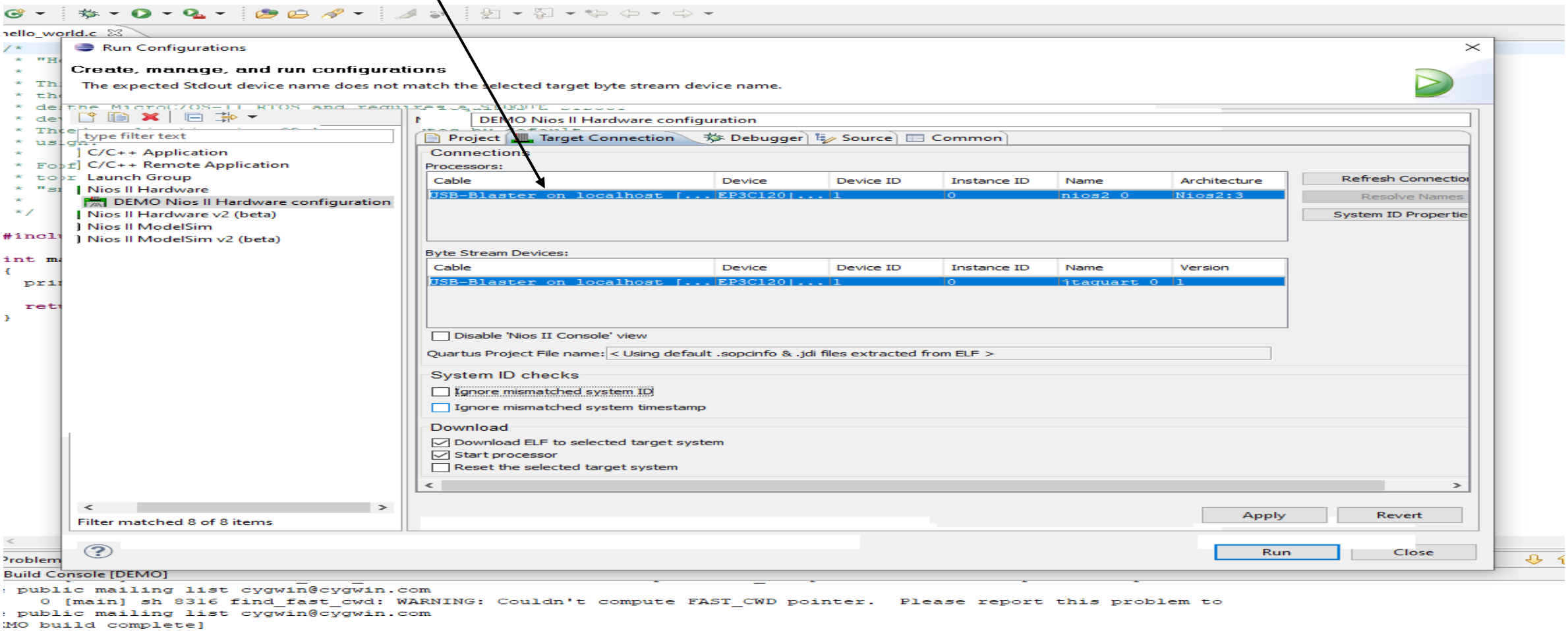
21. In NIOS II window, right click on the project. *Run As → NIOS II Hardware*



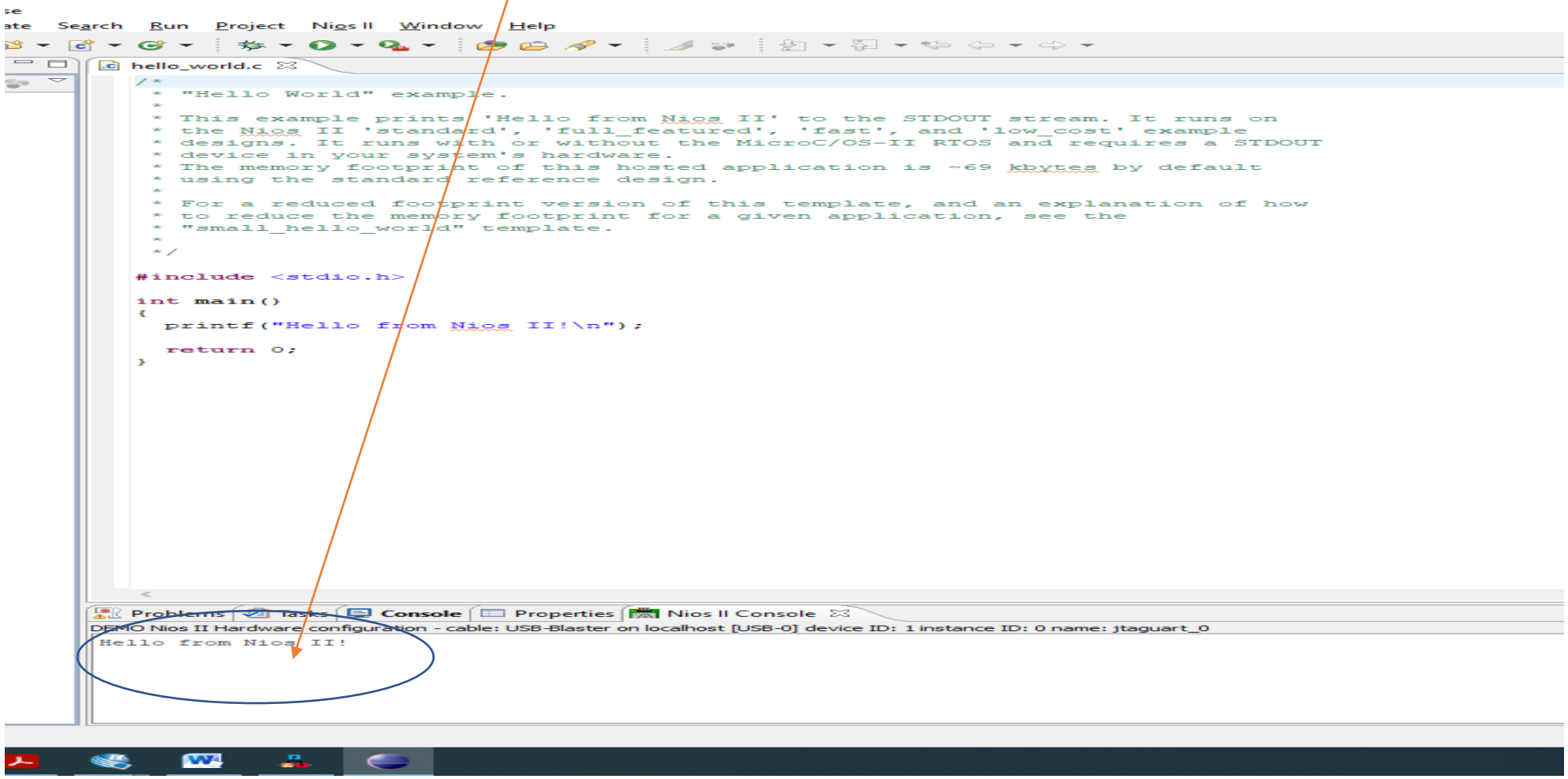
23. Now “Run Configuration” window will pop-up. Move to “Target Connection” tab and click “ Refresh Connections”.



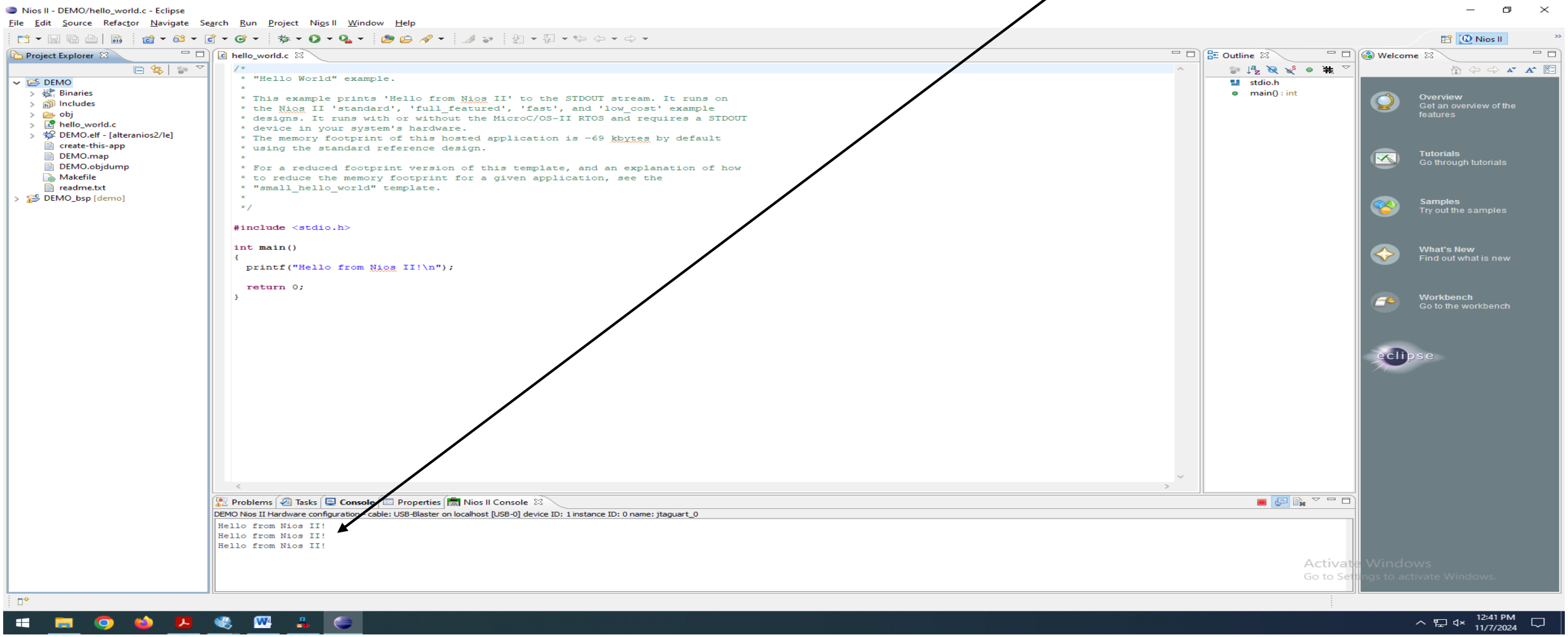
Now “USB-Blaster …..” will get added up



Ouput message “ Hello from Nios II” will appear in the Console window.



Finally, from the hardware when the push button assigned for the *reset_n* is pressed "Hello from Nios II" will again appear in the Console window and repeats for every click.



HAPPY LEARNING!!!!!