

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

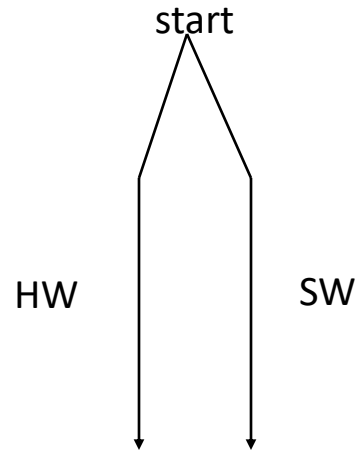
Judged by its objectives in application domain

- Performance
- Design and Manufacturing cost
- Ease of Programmability

It depends on both the hardware and software components

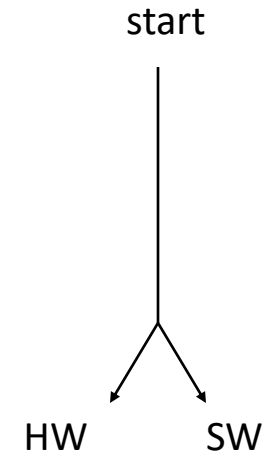
Concurrent design

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.

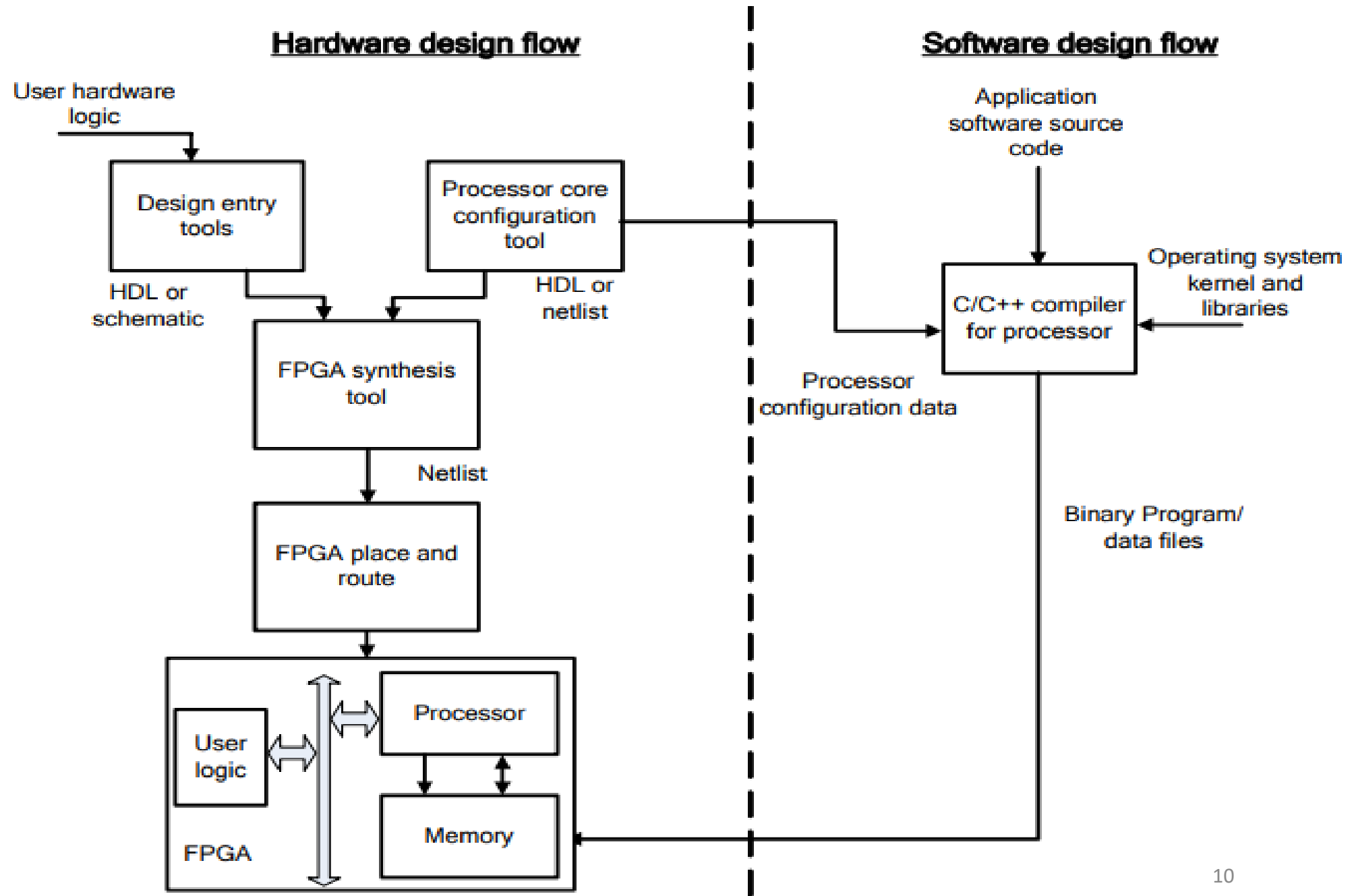
Why co-design?

- **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.*

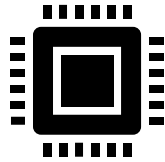
Applications

- 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

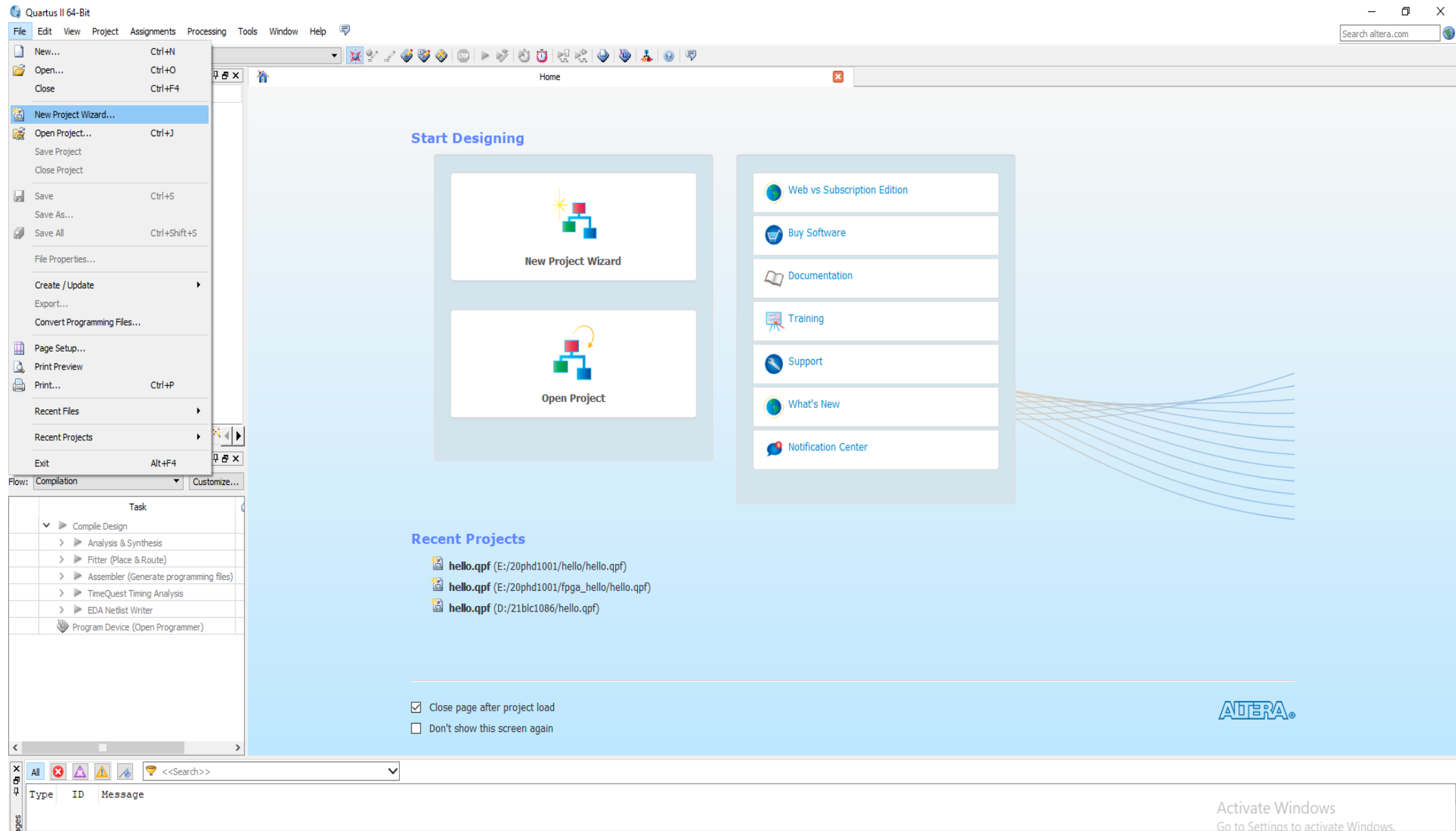
- MATLAB/Simulink:
 - For simulating and modeling system behaviour.
- SystemC:
 - A language for system-level modeling.
- Cadence, Synopsys, Mentor Graphics:
 - Electronic Design Automation (EDA) tools that provide solutions for codesign, including verification and simulation environments.

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*.

New Project Wizard
Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

Device family
Family: *Cyclone IV E*
Devices: *All*

Target device
☐ Auto device selected by the Fitter
☒ Specific device selected in 'Available devices' list
☐ Other: n/a

Show in 'Available devices' list
Package: *FBGA*
Pin count: *780*
Speed grade: *7*
Name filter:
☒ Show advanced devices

Available devices:

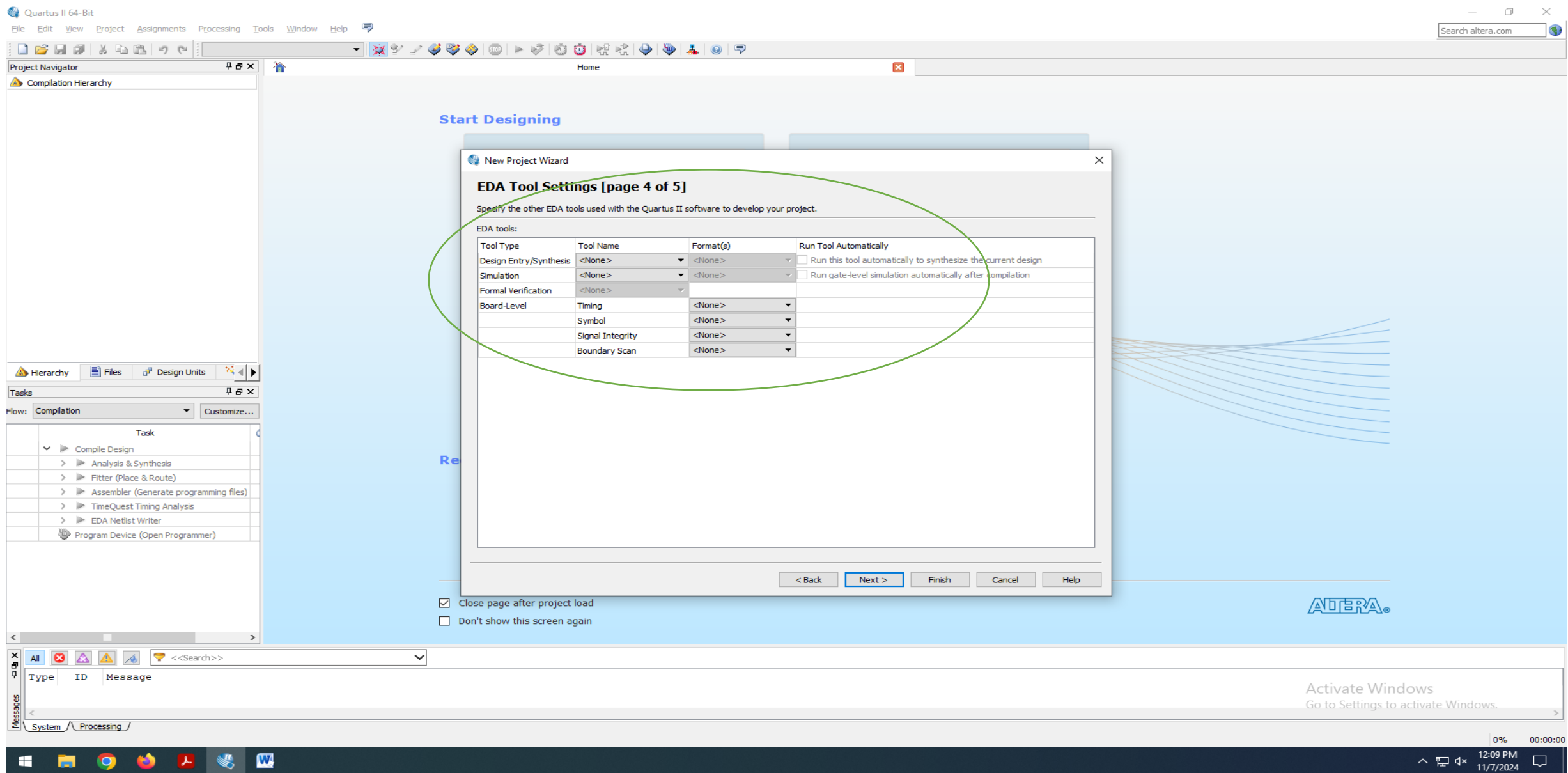
Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded multiplier 9-bit elements	PLL	Glob
EP4CE30F29C7	1.2V	28848	533	608256	132	4	20
EP4CE30F2917	1.2V	28848	533	608256	132	4	20
EP4CE40F29C7	1.2V	39600	533	1161216	232	4	20
EP4CE40F2917	1.2V	39600	533	1161216	232	4	20
EP4CE55F29C7	1.2V	55856	375	2396160	308	4	20
EP4CE55F2917	1.2V	55856	375	2396160	308	4	20
EP4CE75F29C7	1.2V	75408	427	2810880	400	4	20
EP4CE75F2917	1.2V	75408	427	2810880	400	4	20
EP4CE115F29C7	1.2V	114480	529	3981312	532	4	20
EP4CE115F2917	1.2V	114480	529	3981312	532	4	20

☒ Close page after project load
☐ Don't show this screen again

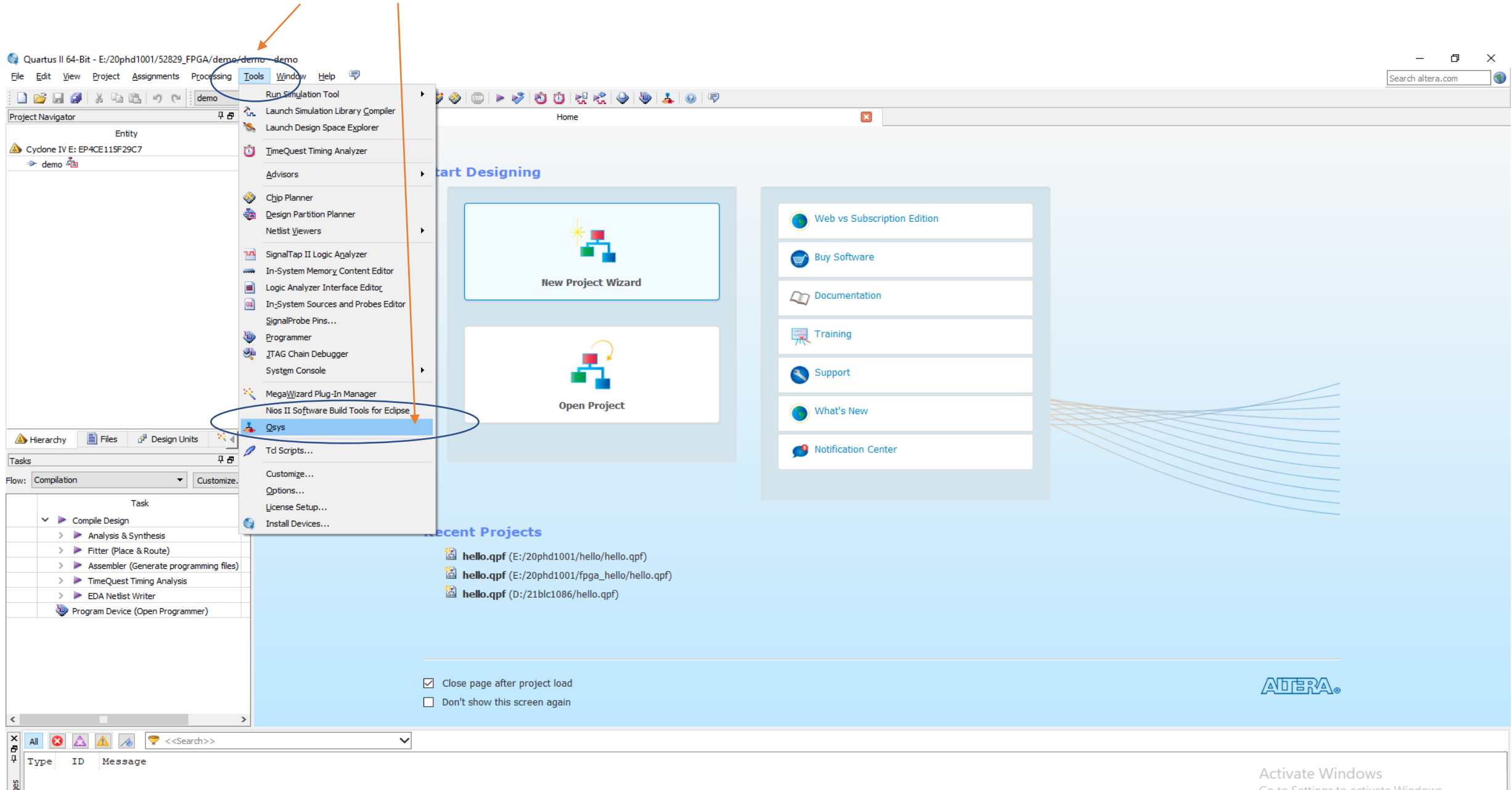
Altera logo

Activate Windows
Go to Settings to activate Windows.

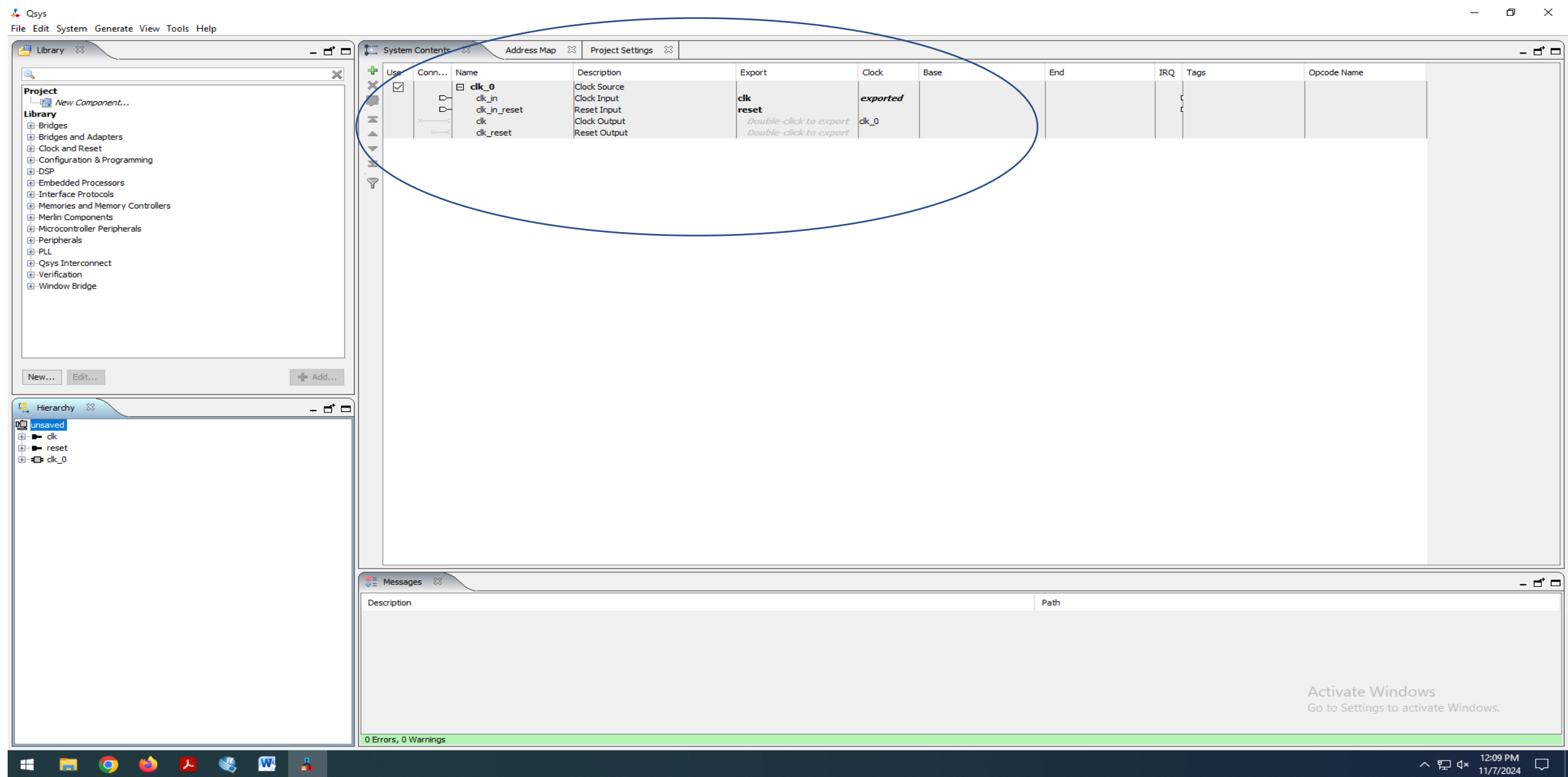
3. Specify all EDA tools as “None”. Finish setup.



4. Open Qsys tool . *Tools* → *Qsys*



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



6. Now we build a system that will display “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

Nios II Processor
altera_nios2_qsys

Block Diagram

clk: clock, reset_n: reset, d_irq: interrupt, jtag_debug_module: avalon, nios2_qsys_0: nios2_qsys_0, data_master: data_master, instruction_master: instruction_master, jtag_debug_module_reset: jtag_debug_module_reset, custom_instruction_master: custom_instruction_master, altera_nios2_qsys: altera_nios2_qsys

Select a Nios II Core

Nios II Core: ☒ Nios II/e, ☐ Nios II/s, ☐ Nios II/f

	Nios II/e	Nios II/s	Nios II/f
Nios II Selector Guide	RISC 32-bit	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dynamic Branch Prediction
Memory Usage (e.g Stratix IV)	Two M9Ks (or equiv.)	Two M9Ks + cache	Three M9Ks + cache

Hardware Arithmetic Operation

Hardware multiplication type: Embedded Multipliers

☐ Hardware divide

Reset Vector

Reset vector memory: None

Reset vector offset: 0x00000000

Reset vector: 0x00000000

Exception Vector

Exception vector memory: None

Exception vector offset: 0x00000020

Exception vector: 0x00000000

Error: nios2_qsys_0: Reset slave is not specified. Please select the reset slave
Error: nios2_qsys_0: Exception slave is not specified. Please select the exception slave

4 Errors, 0 Warnings

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

The screenshot displays the Qsys IDE interface. The **System Contents** window is the central focus, showing a table of system components. A blue circle highlights the **Connections** column, which contains a schematic diagram of the system blocks and their interconnections. Three orange arrows point from text labels to specific blocks in the table:

- On-chip memory** points to the **onchip_memory2_0** block.
- JTAG UART** points to the **jtag_uart_0** block.
- NIOS II processor** points to the **nios2_qsys_0** block.

The table in the **System Contents** window has the following columns: Use, Connections, Name, Description, Export, Clock, Base, End, IRQ, Tags, and Opcode Name.

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
<input checked="" type="checkbox"/>		clk_0	Clock Source	clk	exported					
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input	Double-click to export	clk_0					
<input checked="" type="checkbox"/>		clk	Clock Output	Double-click to export						
<input checked="" type="checkbox"/>		clk_reset	Reset Output	Double-click to export						
<input checked="" type="checkbox"/>		nios2_qsys_0	Nios II Processor	Double-click to export	clk_0					
<input checked="" type="checkbox"/>		clk	Clock Input	Double-click to export	[clk]					
<input checked="" type="checkbox"/>		reset_n	Reset Input	Double-click to export	[clk]					
<input checked="" type="checkbox"/>		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]			IRQ 0	IRQ 31	
<input checked="" type="checkbox"/>		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]					
<input checked="" type="checkbox"/>		jtag_debug_module_r...	Reset Output	Double-click to export	[clk]					
<input checked="" type="checkbox"/>		jtag_debug_module	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0800	0x0fff			
<input checked="" type="checkbox"/>		custom_instruction_m...	Custom Instruction Master	Double-click to export	[clk]					
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or ROM)	Double-click to export	clk_0					
<input checked="" type="checkbox"/>		clk1	Clock Input	Double-click to export	[clk1]					
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]					
<input checked="" type="checkbox"/>		reset1	Reset Input	Double-click to export	[clk1]					
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART	Double-click to export	clk_0					
<input checked="" type="checkbox"/>		clk	Clock Input	Double-click to export	[clk]					
<input checked="" type="checkbox"/>		reset	Reset Input	Double-click to export	[clk]					
<input checked="" type="checkbox"/>		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]					

The **Messages** window at the bottom shows the following errors:

- Reset slave is not specified. Please select the reset slave (System.nios2_qsys_0)
- Exception slave is not specified. Please select the exception slave (System.nios2_qsys_0)
- clk_0.clk_in_reset must be connected to a reset source (System.clk_0)
- nios2_qsys_0.reset_n must be connected to a reset source (System.nios2_qsys_0)
- onchip_memory2_0.reset1 must be connected to a reset source (System.onchip_memory2_0)
- jtag_uart_0.reset must be connected to a reset source (System.jtag_uart_0)

There are also 3 warnings and a total of 6 Errors, 3 Warnings.

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.....”

Qsys - demo.qsys* (E:\20phd1001\52829_FPGA\demo\demo.qsys)

File Edit System Generate View Tools Help

Library

Project

Library

Block Diagram

nios2_qsys_0

clk clock data_master

reset_n reset instruction_master

d_irq interrupt jtag_debug_module_reset

jtag_debug_module reset custom_instruction_master

altera_nios2_qsys

Nios II Processor

altera_nios2_qsys

Core Nios II Caches and Memory Interfaces Advanced Features MMU and MPU Settings JTAG Debug Module

Select a Nios II Core

Nios II Core:

☒ Nios II/e

☐ Nios II/s

☐ Nios II/f

	Nios II/e	Nios II/s	Nios II/f
Nios II Selector Guide	RISC 32-bit	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dynamic Branch Prediction
Memory Usage (e.g Stratix IV)	Two M9Ks (or equiv.)	Two M9Ks + cache	Three M9Ks + cache

Hardware Arithmetic Operation

Hardware multiplication type: Embedded Multipliers

☐ Hardware divide

Reset Vector

Reset vector memory: onchip_memory2_0.s1

Reset vector offset: 0x00000000

Reset vector: 0x00000000

Exception Vector

Exception vector memory: onchip_memory2_0.s1

Exception vector offset: 0x00000020

Exception vector: 0x00000020

1 Warning

Interrupt sender jtag_uart_0_irq is not connected to an interrupt receiver

5 Errors, 1 Warning

System.jtag_uart_0

Cancel Finish

Go to Settings to activate Windows.

12:15 PM 11/7/2024

10. Save the design and Generate HDL.

File → Save → <filename.qsys>

System → Assign Base Addresses

Generate → Generate..

The screenshot displays the Qsys IDE interface. The 'Generate' menu is highlighted in the top menu bar. The 'Address Map' window is open, showing a table of system components and their addresses. The 'Library' pane on the left shows the project hierarchy, and the 'Hierarchy' pane at the bottom shows the component tree. The 'Messages' pane at the bottom indicates 0 Errors, 0 Warnings.

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
<input checked="" type="checkbox"/>		clk_0 clk_in clk_in_reset clk clk_reset	Clock Source Clock Input Reset Input Clock Output Reset Output	clk <i>Double-click to export</i> <i>Double-click to export</i> <i>Double-click to export</i>	exported clk_0					
<input checked="" type="checkbox"/>		nios2_qsys_0 clk reset_n data_master instruction_master jtag_debug_module_r... jtag_debug_module custom_instruction_m...	Nios II Processor Clock Input Reset Input Avalon Memory Mapped Master Avalon Memory Mapped Master Reset Output Avalon Memory Mapped Slave Custom Instruction Master	<i>Double-click to export</i> <i>Double-click to export</i> <i>Double-click to export</i> <i>Double-click to export</i> <i>Double-click to export</i> <i>Double-click to export</i> <i>Double-click to export</i>	clk_0 [clk] [clk] [clk] [clk] [clk]		IRQ 0	IRQ 31		
<input checked="" type="checkbox"/>		onchip_memory2_0 clk1 s1 reset1	On-Chip Memory (RAM or ROM) Clock Input Avalon Memory Mapped Slave Reset Input	<i>Double-click to export</i> <i>Double-click to export</i> <i>Double-click to export</i>	clk_0 [clk1] [clk1]	0x0080_0800	0x0080_0fff			
<input checked="" type="checkbox"/>		jtag_uart_0 clk reset	JTAG UART Clock Input Reset Input	<i>Double-click to export</i> <i>Double-click to export</i>	clk_0 [clk]					
		avalon_jtag_slave	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clk]	0x0080_1000	0x0080_1007			

0 Errors, 0 Warnings

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

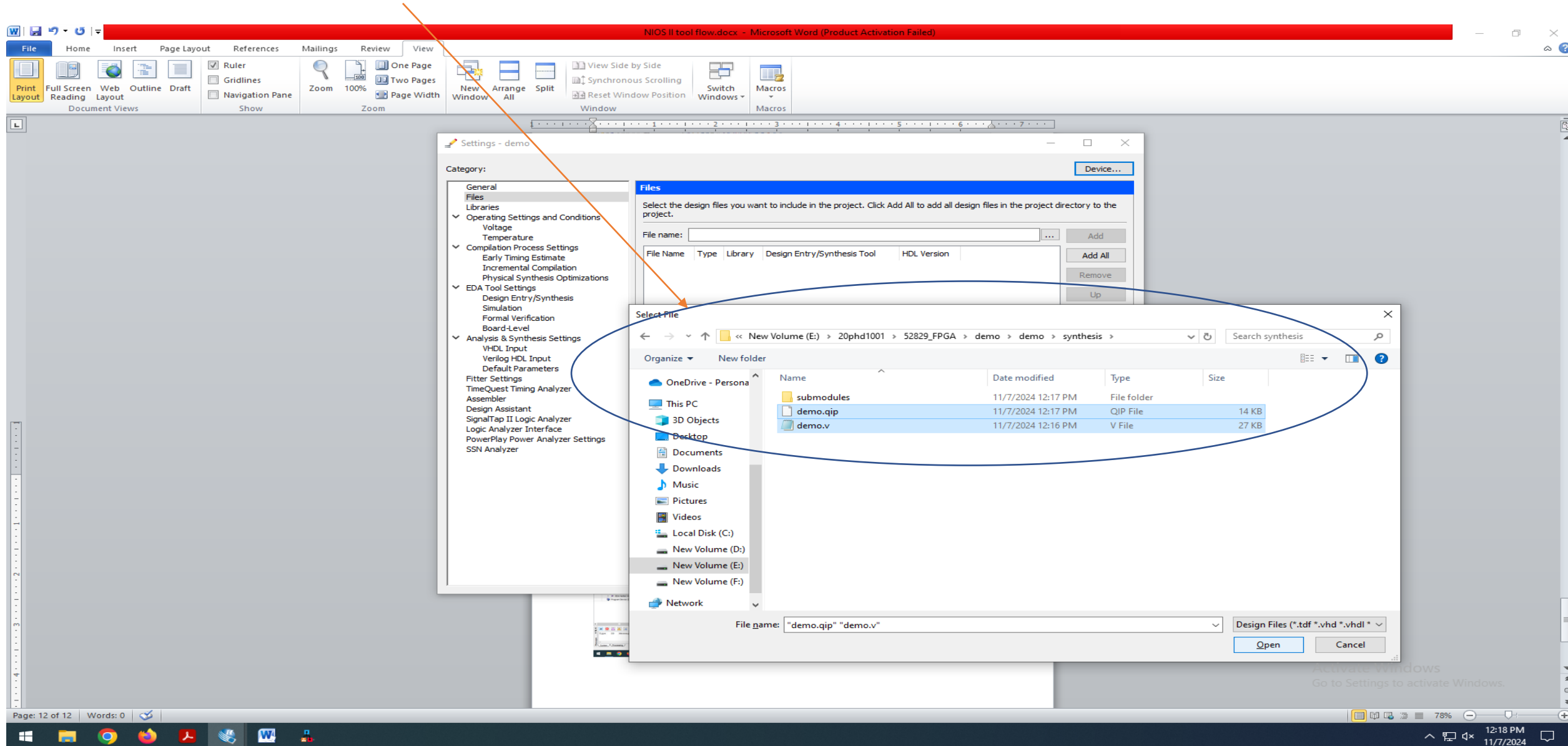
The screenshot shows the Qsys Generation dialog box, which is used to generate various files for a Qsys project. The dialog is divided into several sections: Simulation, Testbench System, Synthesis, and Output Directory. The Output Directory section is highlighted with a blue oval, and an orange arrow points to the 'Path' field within this section. The 'Path' field contains the text 'E:/20phd1001/52829_FPGA/demo/demo'. The 'Simulation' section has 'Create simulation model:' set to 'None'. The 'Testbench System' section has 'Create testbench Qsys system:' set to 'None'. The 'Synthesis' section has 'Create HDL design files for synthesis:' set to 'Verilog' and 'Create block symbol file (.bsf)' checked. The 'Generate' button is visible at the bottom right of the dialog.

Name	Description	Export	Clock	Base	End
clk_0	Clock Source	clk	exported		
clk_in	Clock Input				
clk_in_reset	Reset Input				
clk	Clock Output		clk_0		
clk_reset	Reset Output				
nios2_qsys_0	Nios II Processor				
clk	Clock Input		clk_0		

“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

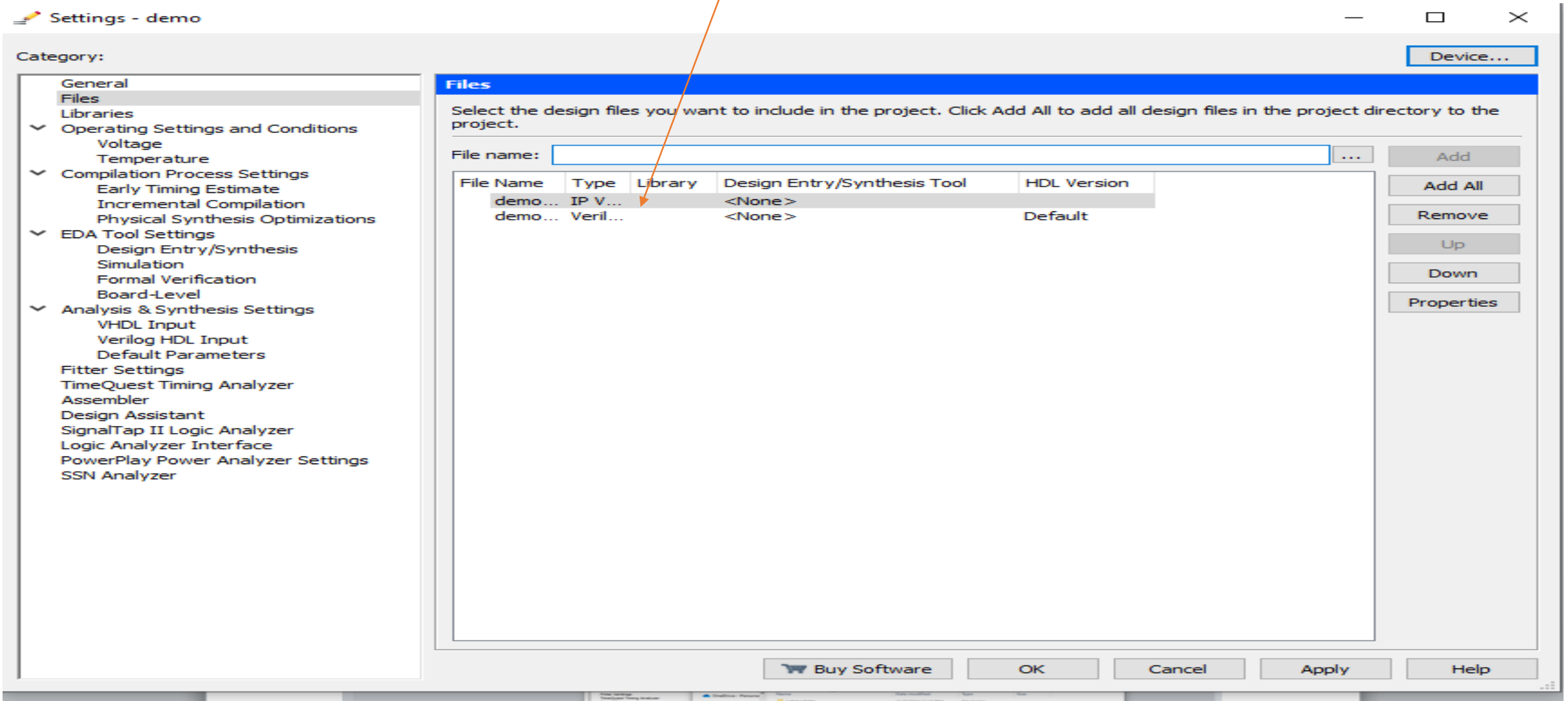


13. Arrange the files.

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*.

The screenshot displays the Qsys IDE interface. The **Tools** menu is open, showing options: **Nios II Software Build Tools for Eclipse**, **Nios II Command Shell [gcc4]**, **System Console**, and **Options...**. The **System Contents** tab is active, showing a table of components and their properties.

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
<input checked="" type="checkbox"/>		clk_0 clk_in clk_in_reset clk clk_reset	Clock Source Clock Input Reset Input Clock Output Reset Output	clk reset Double-click to export Double-click to export	exported clk_0					
<input checked="" type="checkbox"/>		nios2_qsys_0 clk reset_n data_master instruction_master jtag_debug_module_f... jtag_debug_module custom_instruction_m...	Nios II Processor Clock Input Reset Input Avalon Memory Mapped Master Avalon Memory Mapped Master Reset Output Avalon Memory Mapped Slave Custom Instruction Master	Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export	clk_0 [clk] [clk] [clk] [clk]		IRQ 0	IRQ 31		
<input checked="" type="checkbox"/>		onchip_memory2_0 clk1 s1 reset1	On-Chip Memory (RAM or ROM) Clock Input Avalon Memory Mapped Slave Reset Input	Double-click to export Double-click to export Double-click to export	clk_0 [clk1] [clk1]	0x0080_0800	0x0080_0fff			
<input checked="" type="checkbox"/>		jtag_uart_0 clk reset avalon_jtag_slave	JTAG UART Clock Input Reset Input Avalon Memory Mapped Slave	Double-click to export Double-click to export Double-click to export	clk_0 [clk] [clk]	0x0040_0000	0x004e_1a7f			
						0x0080_1000	0x0080_1007			

The **Hierarchy** pane on the left shows the project structure, with **onchip_memory2_0** selected. The **Messages** pane at the bottom shows 0 Errors, 0 Warnings. An "Activate Windows" watermark is visible in the bottom right corner.

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

Qsys - demo.qsys (E:\20phd1001\52829_FPGA\demo\demo.qsys)

File Edit System Generate View Tools Help

Library

Project

New Component...

Library

Bridges

Memory-Mapped

JTAG to Avalon Master Bridge

Interface Protocols

Serial

Avalon-ST JTAG Interface

JTAG UART

Verification

Debug & Performance

Altera Soft Core JTAG IO

JTAG Debug Link

New... Edit... Add...

Hierarchy

demo

clk

reset

clk_0

clk

clk_in

clk_in_reset

clk_reset

jtag_uart_0

avalon_jtag_slave

irq

reset

nios2_qsys_0

clk

custom_instruction_master

d_irq

data_master

instruction_master

jtag_debug_module

jtag_debug_module_reset

reset_n

onchip_memory2_0

clk1

reset1

s1

Connections

Use

Connections

Name

Description

Export

Clock

Base

End

IRQ

Tags

Opcode Name

clk_0

clk_in

clk_in_reset

clk_reset

nios2_qsys_0

clk

reset_n

data_master

instruction_master

jtag_debug_module_r...

jtag_debug_module

custom_instruction_m...

onchip_memory2_0

On-Chip Memory (RAM or ROM)

Workspace Launcher

Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: E:\20phd1001\52829_FPGA\demo

Browse...

Use this as the default and do not ask again

OK

Cancel

Messages

Description

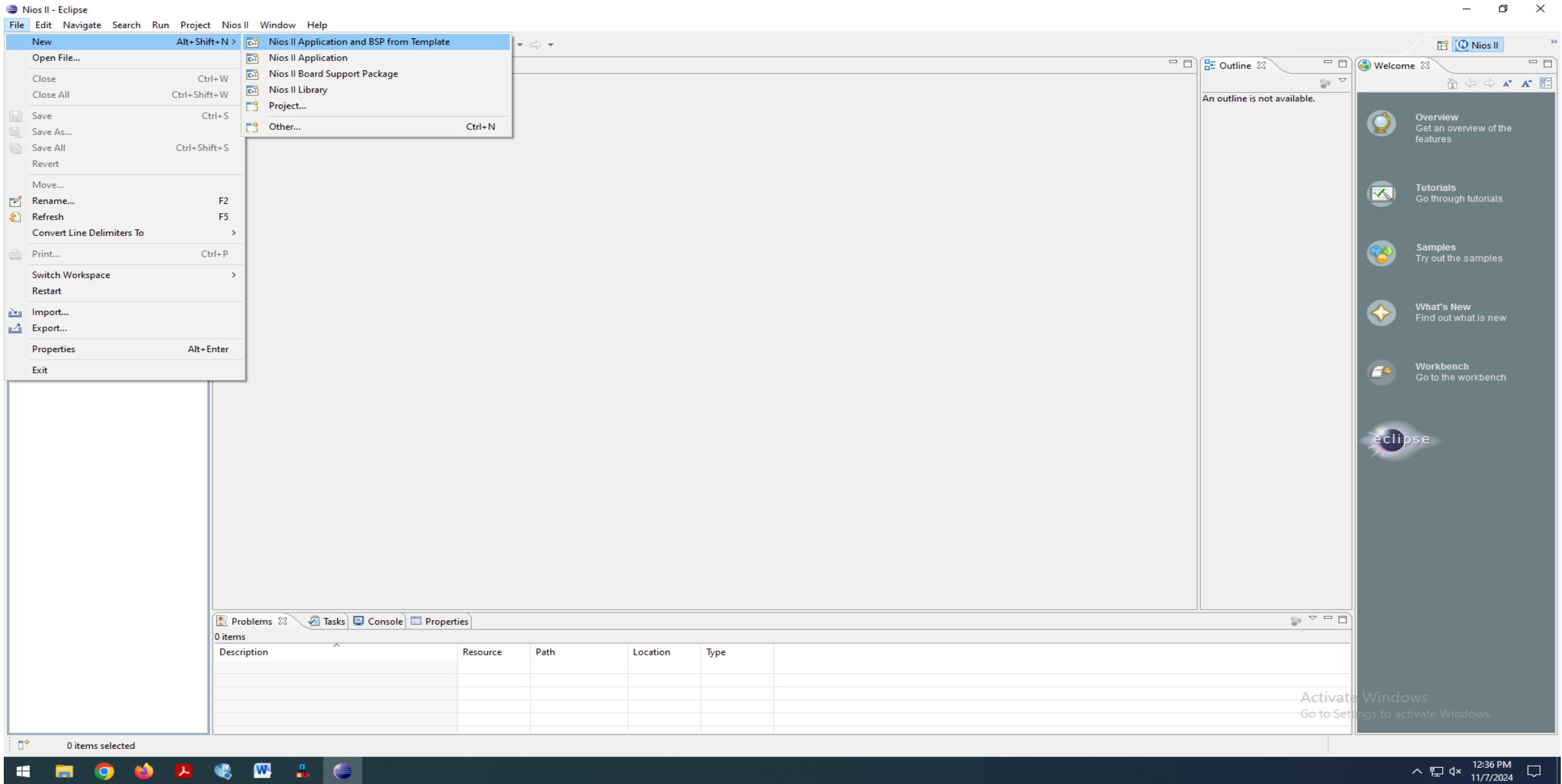
Path

0 Errors, 0 Warnings

Activate Windows
Go to Settings to activate Windows.

12:35 PM
11/7/2024

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.

The screenshot shows the Nios II Eclipse IDE interface. The 'Nios II Application and BSP from Template' wizard is open, displaying the 'Project template' section where 'Hello World' is selected. An 'Open' file dialog is also open, showing the file 'demo.sopcinfo' selected in the 'demo' directory. The 'Project name' field in the wizard is empty, and the 'SOPC Information File name' field is also empty. The 'File name' field in the 'Open' dialog is set to 'demo.sopcinfo'.

Give a project name “Hello”

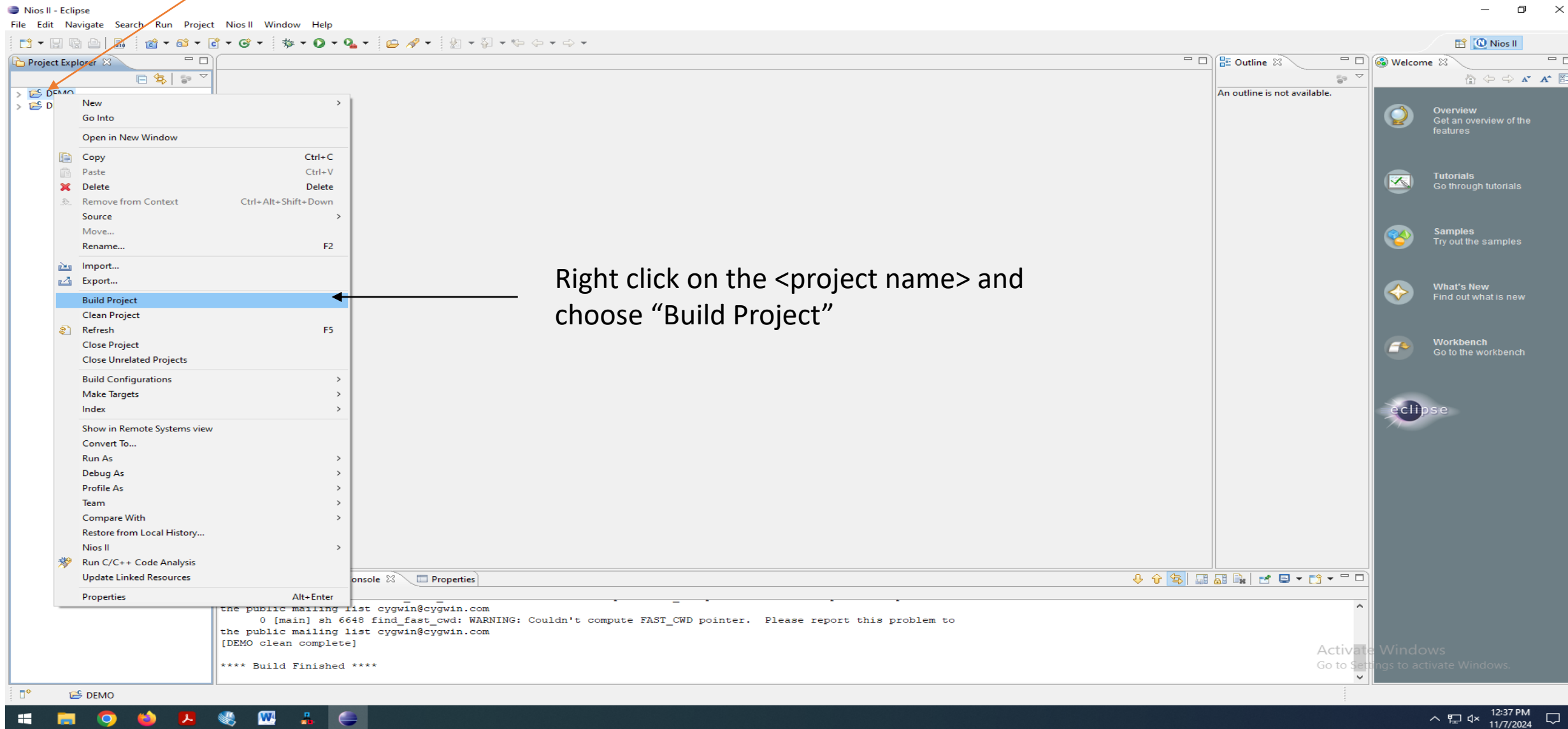
Choose the project template

Activate Windows
Go to Settings to activate Windows.

Name	Date modified	Type	Size
.metadata	11/7/2024 12:36 PM	File folder	
.qsys_edit	11/7/2024 12:14 PM	File folder	
db	11/7/2024 12:34 PM	File folder	
demo	11/7/2024 12:16 PM	File folder	
incremental_db	11/7/2024 12:21 PM	File folder	
output_files	11/7/2024 12:34 PM	File folder	
demo.sopcinfo	11/7/2024 12:33 PM	SOPCINFO File	153 KB

```
Nios II Software Build Tools
For information about how this software example relates to Nios II hardware design examples,
refer to the Design Examples page of the Nios II documentation available with your installation at:
<installation_directory>/nios2eds/documents/index.htm.
</nios2-swexample>
```

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



19. Now the project is built.

Under project required files will be added.

```
/* "Hello World" example.
 *
 * This example prints 'Hello from Nios II' to the STDOUT stream. It runs on
 * the Nios II 'standard', 'full_featured', 'fast', and 'low_cost' example
 * designs. It runs with or without the MicroC/OS-II RTOS and requires a STDOUT
 * device in your system's hardware.
 * The memory footprint of this hosted application is ~69 kbytes by default
 * using the standard reference design.
 *
 * For a reduced footprint version of this template, and an explanation of how
 * to reduce the memory footprint for a given application, see the
 * "small_hello_world" template.
 */

#include <stdio.h>

int main()
{
    printf("Hello from Nios II!\n");

    return 0;
}
```

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

```
CDT Build Console [DEMO]
the public mailing list cygwin@cygwin.com
0 [main] sh 2884 find_fast_cwd: WARNING: Couldn't compute FAST_CWD pointer. Please report this problem to
the public mailing list cygwin@cygwin.com
[DEMO build complete]

**** Build Finished ****
```

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

The screenshot shows the Quartus II 64-Bit IDE interface. The 'Tools' menu is open, and the 'Programmer' option is highlighted. The main window displays the HDL code for a Nios2 Qsys system, showing various interconnect and onchip memory components. The Messages window at the bottom shows a successful compilation message: "293000 Quartus II Full Compilation was successful. 0 errors, 28 warnings".

Quartus II 64-Bit - E:/20phd1001/52829_FPGA/demo/demo - demo

File Edit View Project Assignments Processing Tools Window Help

Project Navigator

Files

- demo/synthesis/demo.qip
- demo/synthesis/demo.v

Tasks

Flow: Compilation

Task

- Compile Design
- Analysis & Synthesis
- Fitter (Place & Route)
- Assembler (Generate programming files)
- TimeQuest Timing Analysis
- EDA Netlist Writer
- Program Device (Open Programmer)

Tools

- Run Simulation Tool
- Launch Simulation Library Compiler
- Launch Design Space Explorer
- TimeQuest Timing Analyzer
- Advisors
- Chip Planner
- Design Partition Planner
- Netlist Viewers
- SignalTap II Logic Analyzer
- In-System Memory Content Editor
- Logic Analyzer Interface Editor
- In-System Sources and Probes Editor
- SignalProbe Pins...
- Programmer**
- JTAG Chain Debugger
- System Console
- MegaWizard Plug-In Manager
- Nios II Software Build Tools for Eclipse
- Qsys
- Tcl Scripts...
- Customize...
- Options...
- License Setup...
- Install Devices...

demo/synthesis/demo.v

```
CDS version 13.1 162 at 2024.11.07.12:33:32

clk_clk, // clk.clk
reset_reset_n // reset.reset_n

interconnect_0_jtag_uart_0_avalon_jtag_slave_waitrequest; // jtag_uart_0:av_waitrequest -> mm_interconnect_0:jtag_uart_0_avalon_jtag_slave_waitrequest
interconnect_0_jtag_uart_0_avalon_jtag_slave_writedata; // mm_interconnect_0:jtag_uart_0_avalon_jtag_slave_writedata -> jtag_uart_0:av_writedata
interconnect_0_jtag_uart_0_avalon_jtag_slave_address; // mm_interconnect_0:jtag_uart_0_avalon_jtag_slave_address -> jtag_uart_0:av_address
interconnect_0_jtag_uart_0_avalon_jtag_slave_chipselect; // mm_interconnect_0:jtag_uart_0_avalon_jtag_slave_chipselect -> jtag_uart_0:av_chipselect
interconnect_0_jtag_uart_0_avalon_jtag_slave_write; // mm_interconnect_0:jtag_uart_0_avalon_jtag_slave_write -> jtag_uart_0:av_write_n
interconnect_0_jtag_uart_0_avalon_jtag_slave_read; // mm_interconnect_0:jtag_uart_0_avalon_jtag_slave_read -> jtag_uart_0:av_read_n
interconnect_0_jtag_uart_0_avalon_jtag_slave_readdata; // jtag_uart_0:av_readdata -> mm_interconnect_0:jtag_uart_0_avalon_jtag_slave_readdata
interconnect_0_onchip_memory2_0_sl_writedata; // mm_interconnect_0:onchip_memory2_0_sl_writedata -> onchip_memory2_0:writedata
interconnect_0_onchip_memory2_0_sl_address; // mm_interconnect_0:onchip_memory2_0_sl_address -> onchip_memory2_0:address
interconnect_0_onchip_memory2_0_sl_chipselect; // mm_interconnect_0:onchip_memory2_0_sl_chipselect -> onchip_memory2_0:chipselect
interconnect_0_onchip_memory2_0_sl_clk; // mm_interconnect_0:onchip_memory2_0_sl_clk -> onchip_memory2_0:clken
interconnect_0_onchip_memory2_0_sl_write; // mm_interconnect_0:onchip_memory2_0_sl_write -> onchip_memory2_0:write
interconnect_0_onchip_memory2_0_sl_readdata; // mm_interconnect_0:onchip_memory2_0_sl_readdata -> onchip_memory2_0:readdata
interconnect_0_onchip_memory2_0_sl_byteenable; // mm_interconnect_0:onchip_memory2_0_sl_byteenable -> onchip_memory2_0:byteenable
interconnect_0_nios2_qsys_0_jtag_debug_module_waitrequest; // nios2_qsys_0:jtag_debug_module_waitrequest -> mm_interconnect_0:nios2_qsys_0_jtag_debug_module_waitrequest
interconnect_0_nios2_qsys_0_jtag_debug_module_writedata; // mm_interconnect_0:nios2_qsys_0_jtag_debug_module_writedata -> nios2_qsys_0:jtag_debug_module_writedata
interconnect_0_nios2_qsys_0_jtag_debug_module_address; // mm_interconnect_0:nios2_qsys_0_jtag_debug_module_address -> nios2_qsys_0:jtag_debug_module_address
interconnect_0_nios2_qsys_0_jtag_debug_module_write; // mm_interconnect_0:nios2_qsys_0_jtag_debug_module_write -> nios2_qsys_0:jtag_debug_module_write
interconnect_0_nios2_qsys_0_jtag_debug_module_read; // mm_interconnect_0:nios2_qsys_0_jtag_debug_module_read -> nios2_qsys_0:jtag_debug_module_read
interconnect_0_nios2_qsys_0_jtag_debug_module_readdata; // nios2_qsys_0:jtag_debug_module_readdata -> mm_interconnect_0:nios2_qsys_0_jtag_debug_module_readdata
interconnect_0_nios2_qsys_0_jtag_debug_module_debugaccess; // mm_interconnect_0:nios2_qsys_0_jtag_debug_module_debugaccess -> nios2_qsys_0:jtag_debug_module_debugaccess
interconnect_0_nios2_qsys_0_jtag_debug_module_byteenable; // mm_interconnect_0:nios2_qsys_0_jtag_debug_module_byteenable -> nios2_qsys_0:jtag_debug_module_byteenable
nios2_qsys_0_data_master_waitrequest; // mm_interconnect_0:nios2_qsys_0_data_master_waitrequest -> nios2_qsys_0:d_waitrequest
nios2_qsys_0_data_master_writedata; // nios2_qsys_0:d_writedata -> mm_interconnect_0:nios2_qsys_0_data_master_writedata
nios2_qsys_0_data_master_address; // nios2_qsys_0:d_address -> mm_interconnect_0:nios2_qsys_0_data_master_address
nios2_qsys_0_data_master_write; // nios2_qsys_0:d_write -> mm_interconnect_0:nios2_qsys_0_data_master_write
nios2_qsys_0_data_master_read; // nios2_qsys_0:d_read -> mm_interconnect_0:nios2_qsys_0_data_master_read
nios2_qsys_0_data_master_readdata; // nios2_qsys_0:jtag_debug_module_debugaccess_to_roms -> mm_interconnect_0:nios2_qsys_0_data_master_readdata
nios2_qsys_0_data_master_debugaccess; // nios2_qsys_0:d_byteenable -> mm_interconnect_0:nios2_qsys_0_data_master_byteenable
nios2_qsys_0_data_master_byteenable; // mm_interconnect_0:nios2_qsys_0_data_master_byteenable -> mm_interconnect_0:nios2_qsys_0_data_master_byteenable
nios2_qsys_0_instruction_master_waitrequest; // mm_interconnect_0:nios2_qsys_0_instruction_master_waitrequest -> nios2_qsys_0:i_waitrequest
nios2_qsys_0_instruction_master_address; // nios2_qsys_0:i_address -> mm_interconnect_0:nios2_qsys_0_instruction_master_address
nios2_qsys_0_instruction_master_read; // nios2_qsys_0:i_read -> mm_interconnect_0:nios2_qsys_0_instruction_master_read
nios2_qsys_0_instruction_master_readdata; // mm_interconnect_0:nios2_qsys_0_instruction_master_readdata -> nios2_qsys_0:i_readdata
irq_mapper_receiver0_irq; // jtag_uart_0:av_irq -> irq_mapper:receiver0_irq
nios2_qsys_0_d_irq_irq; // irq_mapper:sender_irq -> nios2_qsys_0:d_irq
rst_controller_reset_out_reset; // rst_controller:reset_out -> [irq_mapper:reset, jtag_uart_0:rst_n, mm_interconnect_0:nios2_qsys_0_reset_n] res
```

Messages

293000 Quartus II Full Compilation was successful. 0 errors, 28 warnings

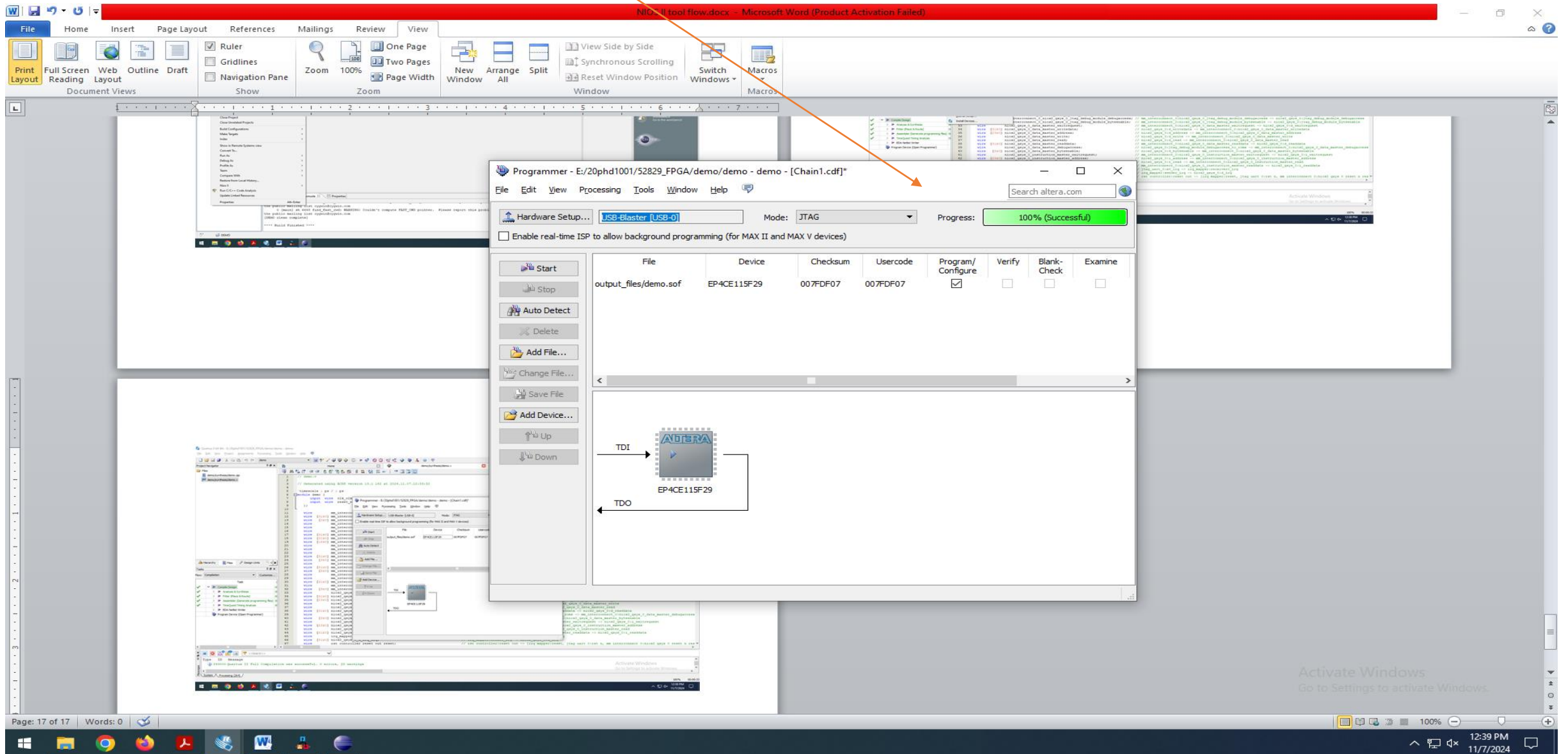
System Processing (264)

Opens a Programmer window

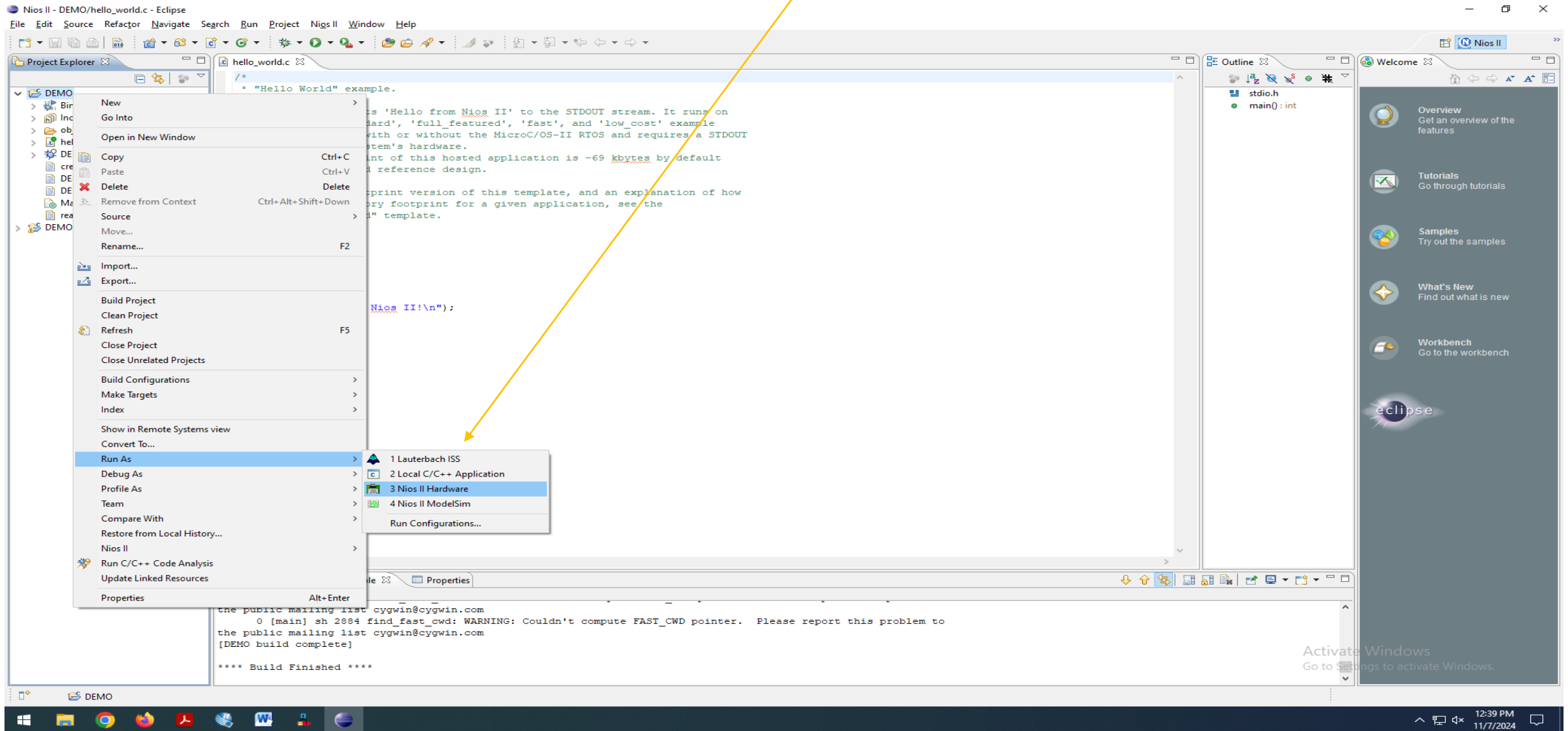
Activate Windows
Go to Settings to activate Windows.

100% 00:00:33
12:38 PM
11/7/2024

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”.

The screenshot shows the Eclipse IDE interface with the 'Run Configurations' dialog box open. The 'Target Connection' tab is selected, showing a table of connections and a 'Refresh Connections' button. The background shows the 'hello_world.c' source file and the build console output.

Run Configurations Dialog:

- Tab: Target Connection
- Buttons: Refresh Connections, Resolve Names, System ID Properties...
- Table 1 (Architecture):

Device ID	Instance ID	Name	Architecture

- Table 2 (Version):

Device ID	Instance ID	Name	Version

Source File (hello_world.c):

```
/* "Hello World" example.
 * This example prints 'Hello from the Nios II 'standard', 'full_featured' designs. It runs with or without the MicroC/OS-II RTOS and requires a STDOUT device in your system's hardware.
 * The memory footprint of this hosted application is ~69 kbuses by default.
 * For a reduced footprint version of the application, use the 'small_hello_world' template.
 */

#include <stdio.h>

int main()
{
    printf("Hello from Nios II!\n");
    return 0;
}
```

Build Console Output:

```
CDT Build Console [DEMO]
the public mailing list cygwin@cygwin.com
0 [main] sh 8316 find_fast_cwd: WARNING: Couldn't compute FAST_CWD pointer. Please report this problem to
the public mailing list cygwin@cygwin.com
[DEMO build complete]

**** Build Finished ****
```

Now “USB-Blaster” will get added up

Run Configurations

Create, manage, and run configurations

The expected Stdout device name does not match the selected target byte stream device name.

DEMO Nios II Hardware configuration

Connections

Processors:

Cable	Device	Device ID	Instance ID	Name	Architecture
USB-Blaster on localhost [... EP3C120 ...	EP3C120 ...	1	0	nios2_0	Nios2:3

Byte Stream Devices:

Cable	Device	Device ID	Instance ID	Name	Version
USB-Blaster on localhost [... EP3C120 ...	EP3C120 ...	1	0	itaguart_0	1

☐ Disable 'Nios II Console' view

Quartus Project File name: < Using default .sopcinfo & .jdi files extracted from ELF >

System ID checks

☐ Ignore mismatched system ID

☐ Ignore mismatched system timestamp

Download

☒ Download ELF to selected target system

☒ Start processor

☐ Reset the selected target system

Filter matched 8 of 8 items

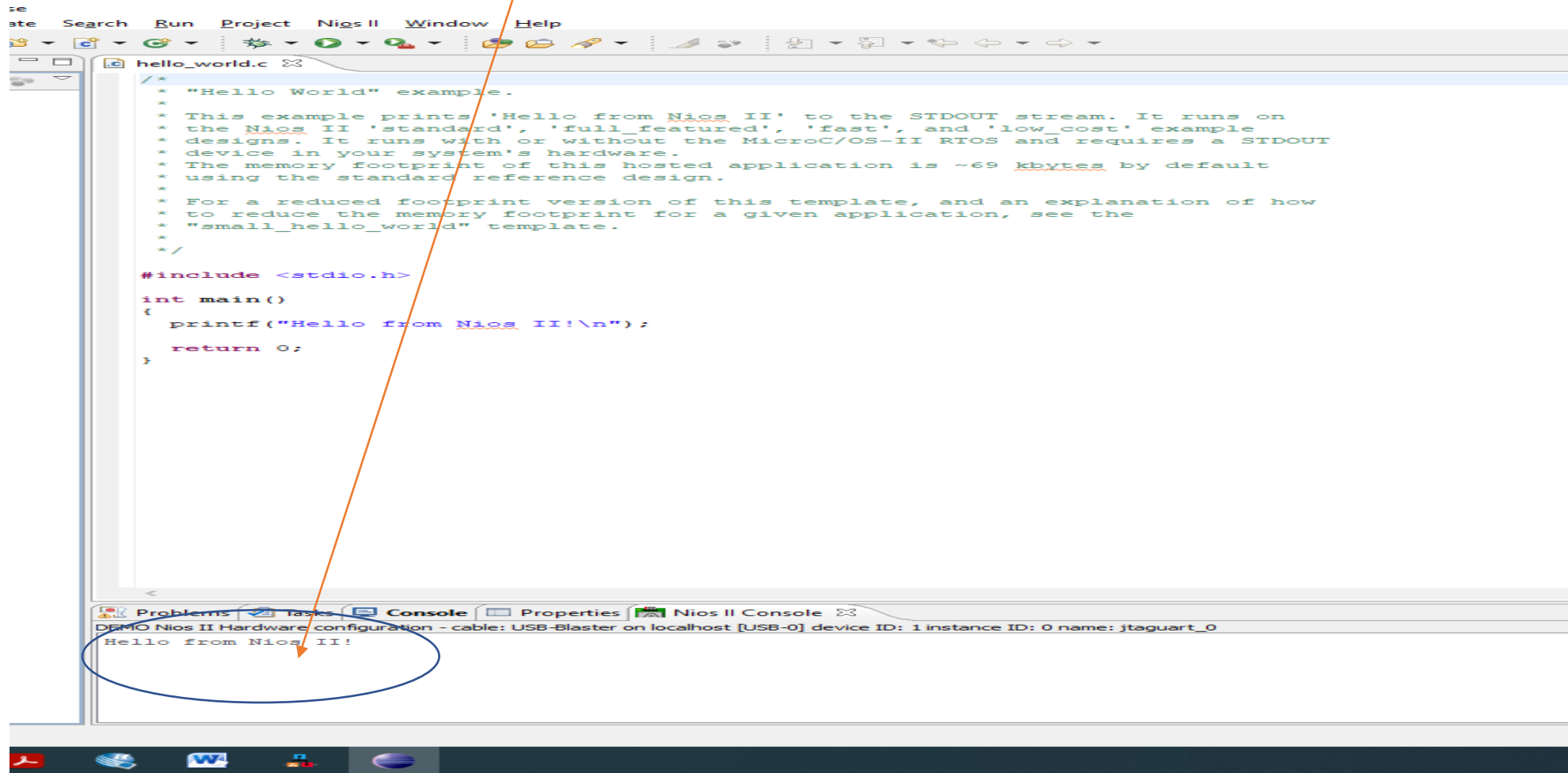
Run

Close

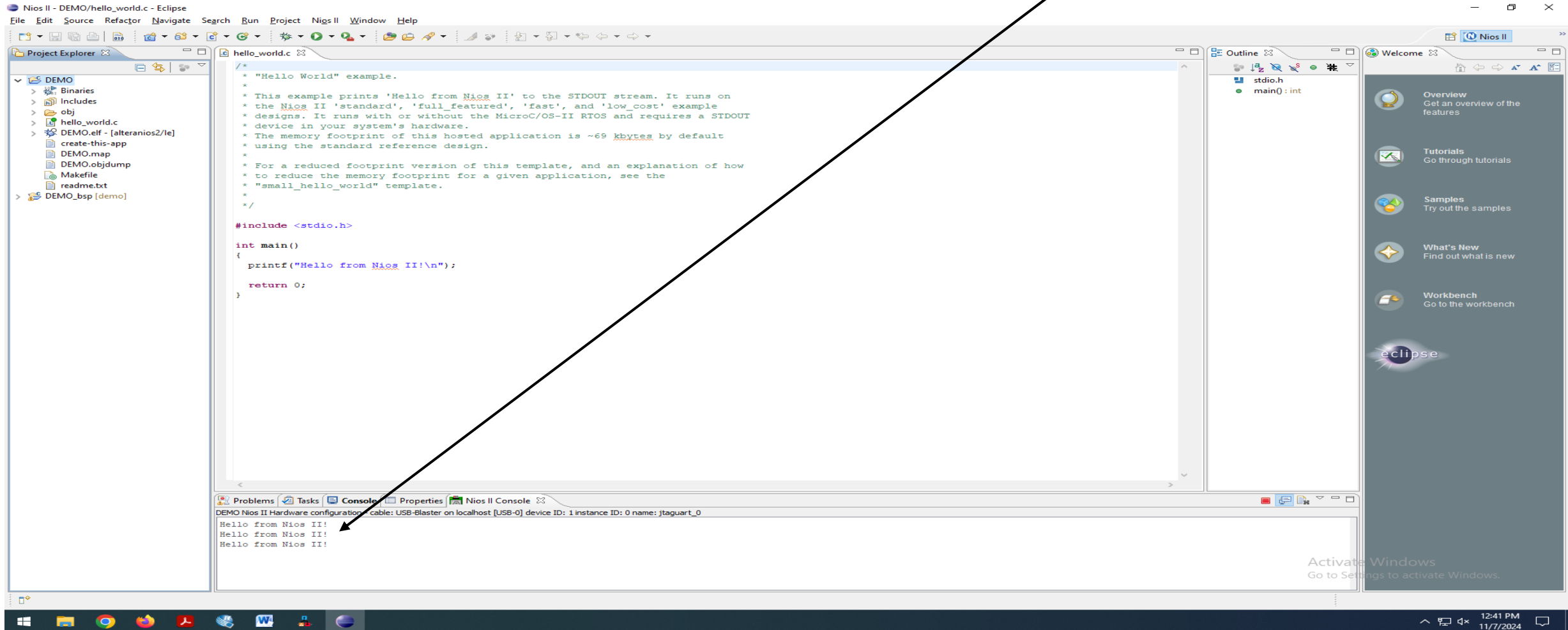
Build Console [DEMO]

```
public mailing list cygwin@cygwin.com
0 [main] sh 8316 find_fast_cwd: WARNING: Couldn't compute FAST_CWD pointer. Please report this problem to
public mailing list cygwin@cygwin.com
MO build complete]
```


Output 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!!!!!!