A Simple Nios® V Implementation on MAX® 10-10M08 Evaluation Kit

By Sean D. Liming and John R. Malin
Annabooks, LLC. – www.annabooks.com

December 2025

The first few articles in the FPGA series covered different Nios® II projects on the MAX 10. Nios II has been around for a long time. With the release of RISC V, Intel developed the Nios V 32-bit soft processor core based on RISC V ISA (RISC V 32I) to replace Nios II moving forward. Nios V was first introduced to the higher-end of the Intel FPGA family. Now that Nios V has been made available in the Quartus Prime Standard and Lite editions, Nios V projects can be built on the Max 10, Cyclone V, Cyclone 10LP, and Cyclone IV. This hands-on article will walk through the creation of a basic Nios V design, and we will explore some tool features along the way.

Please, see the article *Altera™ Quartus® Prime Lite v25.1 and Nios® V Install Instructions* on Annabooks.com for information on how to install the Quartus software needed for this hands-on exercise.

The Project Requirements:

- Quartus Prime Lite Edition V25.1, Ashling RiscFree™ IDE, and Nios® V license are already installed.
- Intel® MAX® 10 - 10M08 Evaluation Kit and the schematic for the evaluation board are required. The schematic PDF file can be downloaded from the Intel FPGA website.
- FPGA Programming cable – USB Blaster II or EthernetBlaster II. The Intel® MAX® 10 - 10M08 Evaluation Kit doesn't have a built-in USB Blaster II onboard.

Note: There are equivalent MAX 10 development and evaluation boards available. These boards can also be used as the target, but you will have to adjust for the available features on the board. Please, make sure that you have the board's schematic files as these will be needed to identify pins.

Note: As of this writing, Altera is still migrating information and licensing details from Intel. You will see a mix of both company names until the transfer is complete.

The Nios V workflow is different than the Nios II projects.

- The first step is still the same: creating the hardware design in Quartus Prime and Platform Builder.
- The second step is to use the Nios V command line to create the BSP and the cmake project.
- The final step is to write and debug the application with RiscFree IDE.

1.1 Part 1: Basic Nios V Design

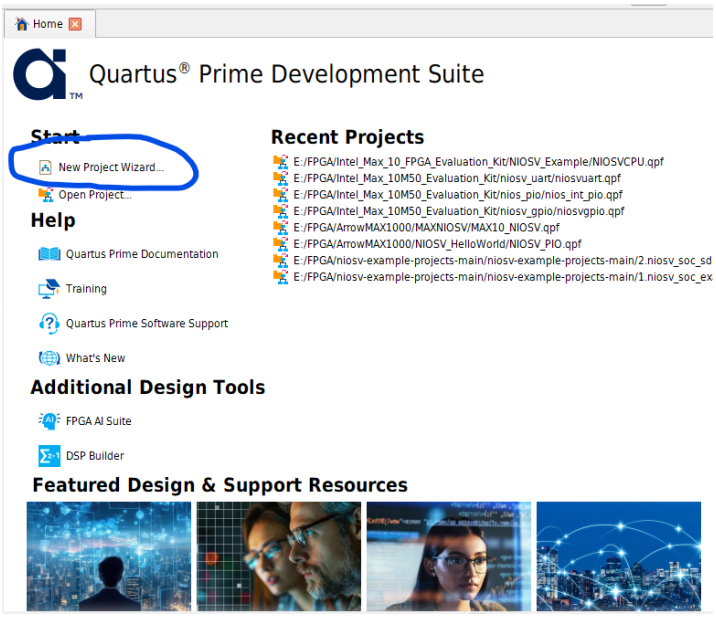
For the FPGA design, we will have a Nios V/m processor IP block, along with an onboard RAM IP Block, and a JTAG UART IP Block. The application will simply send messages out the JTAG interface to a console application.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

1.1.1 Create the Project

The first step is to create a design project.

1. Open Quartus.
2. Click on the New Project Wizard.

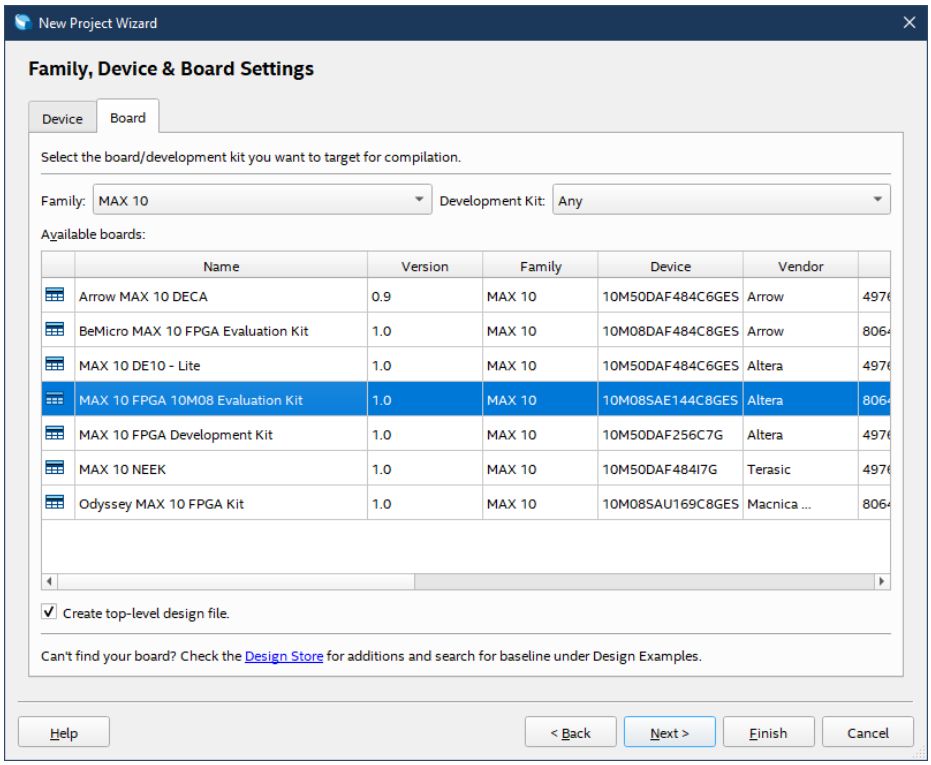


3. Select or create a project directory \NIOSV_Example (**Do not use the Quartus installation directory**) and name the project: “NIOSVCPU”. Click Next.

Note: By default, the root directory is the Quartus installation directory. Make sure the root project directory is a separate path from the Quartus installation files. Also, there can be no spaces in the name of the folders or projects.

4. Project Type: Empty project, click Next.
5. Add File: no files to add, click Next.
6. Family, Device & Board Settings: click the Board tab and select: MAX 10 FPGA 10M08 Evaluation Kit. Click Next.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.



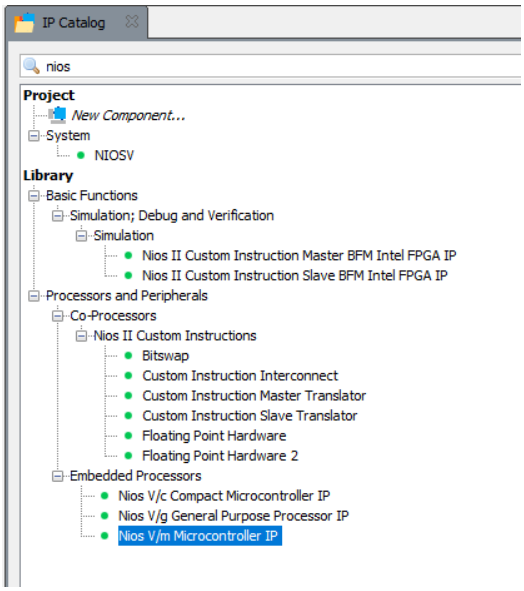
7. EDA Tools: click Next.
8. Summary: click Finish

1.1.2 Create the Design Step 1: Platform Designer

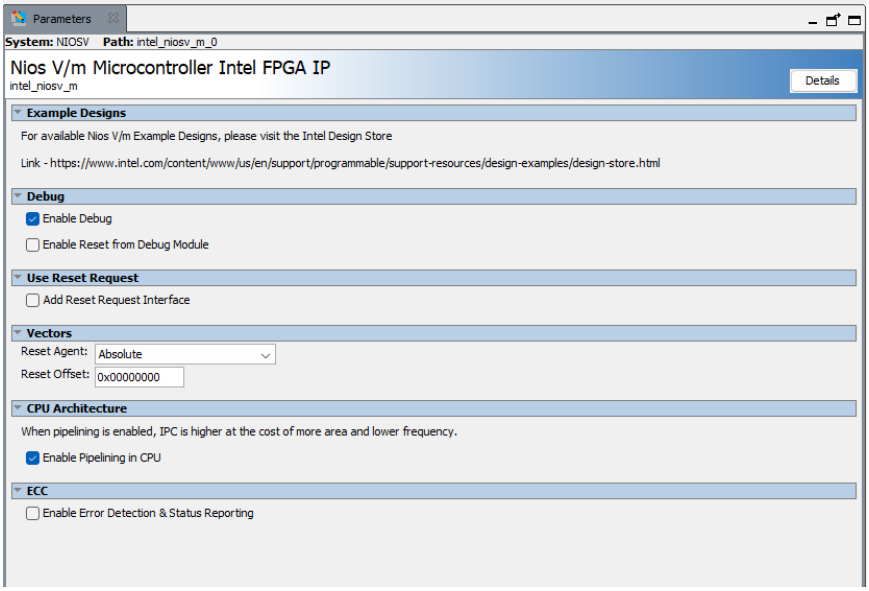
Quartus supports many design types to create an FPGA design. The Platform Designer tool will be used for this hands-on exercise. Platform Builder makes it easy to add already-built IP blocks and interconnect them.

1. From the menu, select Tools->Platform Designer, or the Platform Designer icon from the toolbar. The Platform Designer tool is launched. By default, a clock (clk_0) is added to the design. Platform Designer makes it easy to add IP blocks and make interconnections between the blocks.
2. The top left pane contains the IP Catalog with all the available IP blocks that come with Quartus Prime. In the search box, type nios.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

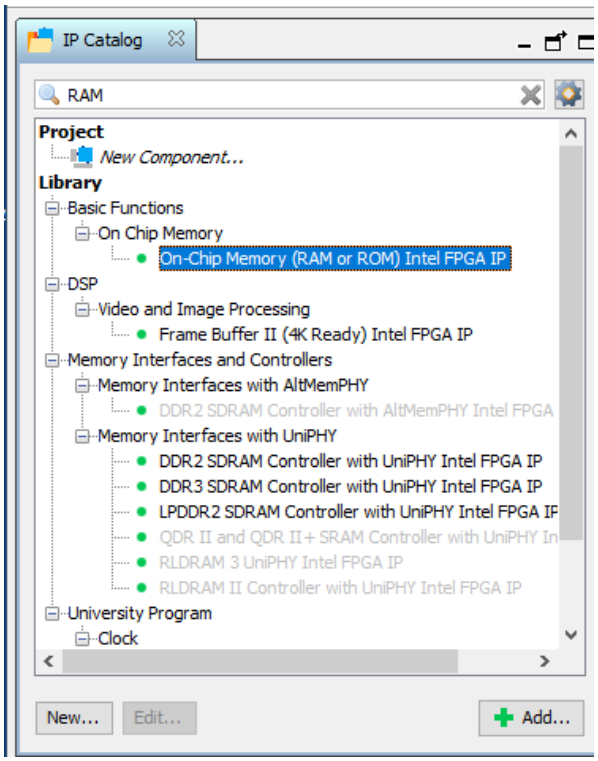


3. Expand the Processors and Peripherals and the Embedded Processors branches. Under Embedded Processors double-click on the Nios V/m Processor Intel FPGA IP.
4. This will open the Nios V/m Configuration page. We will keep the defaults for now. Click Finish.

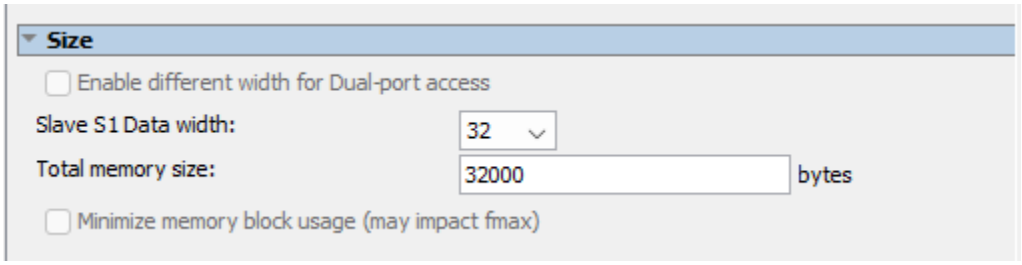


5. Now let's add the RAM IP block. In the IP Catalog enter RAM in the search box.
6. Double-click on On-chip Memory (RAM or ROM) in the Intel FPGA IP.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.



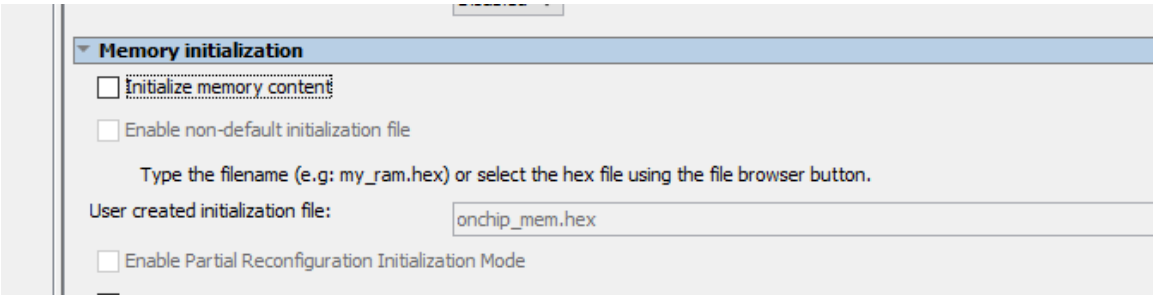
7. The configuration page will appear. Change the Total memory size to 32000. More memory is require to run Nios V applications.



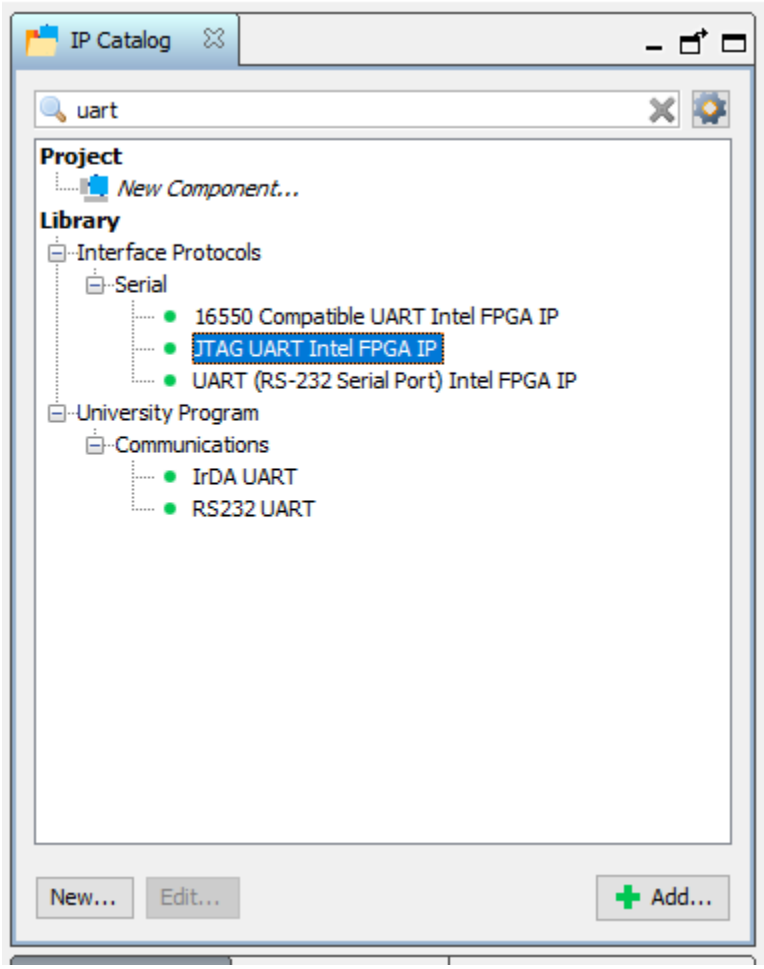
Intel Max 10 Part Number	Logic Elements	Maximum Embedded Memory	Maximum User I/O Count
10M02	2000	108 Kbits	246
10M04	4000	189 Kbits	246
10M08	**8000**	**378 Kbits**	**250**
10M16	16000	549 Kbits	320
10M25	25000	675 Kbits	360
10M40	40000	1.26 Mbits	500
10M50	50000	1.638 Mbits	500

8. Uncheck the box for “Initialize memory content”, and click Finish.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

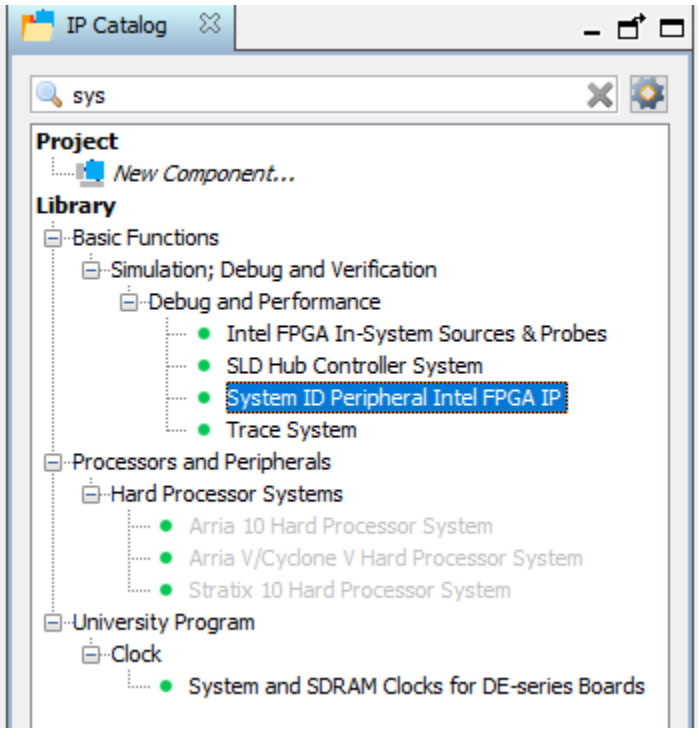


9. In the IP Catalog search, enter uart.
10. Double-click on the JTAG UART Intel FPGA IP.

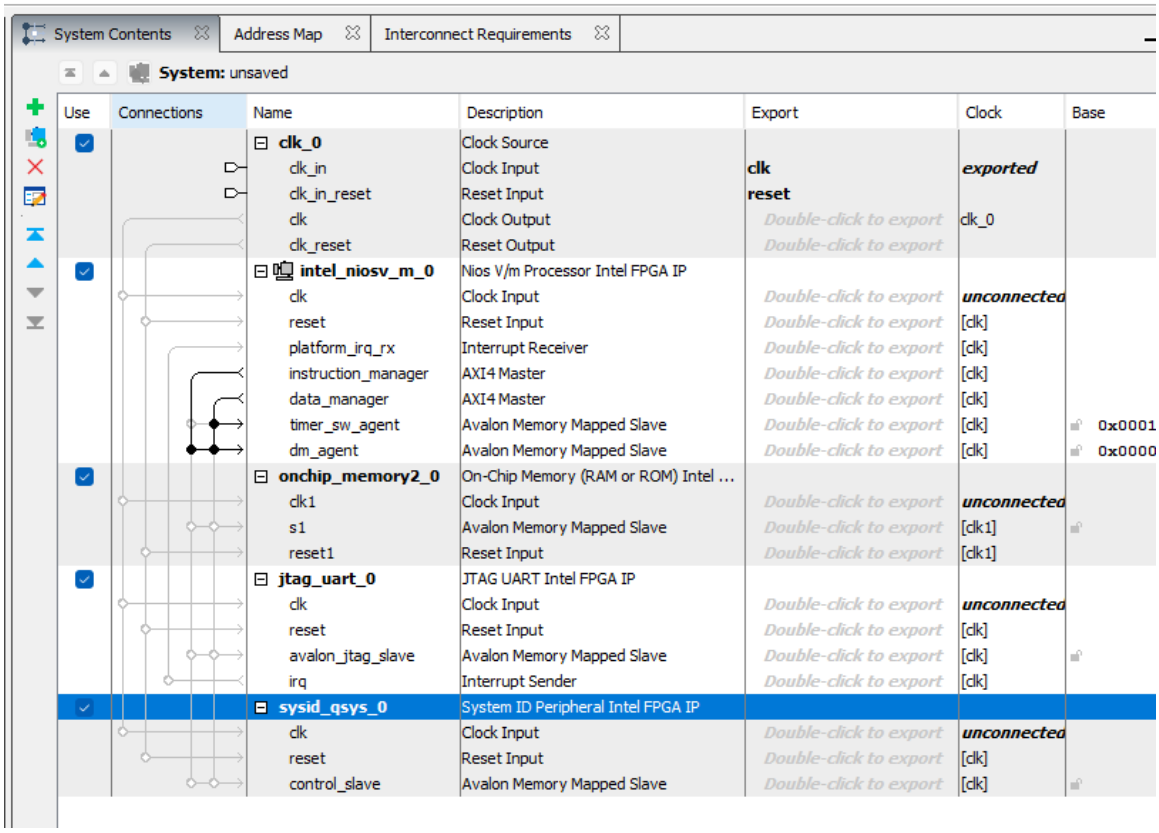


11. A configuration page will appear. There are no changes to be made. Click Finish.
12. In the IP Catalog search, enter the system ID.
13. Double-click on the System ID Peripheral Intel FPGA IP.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

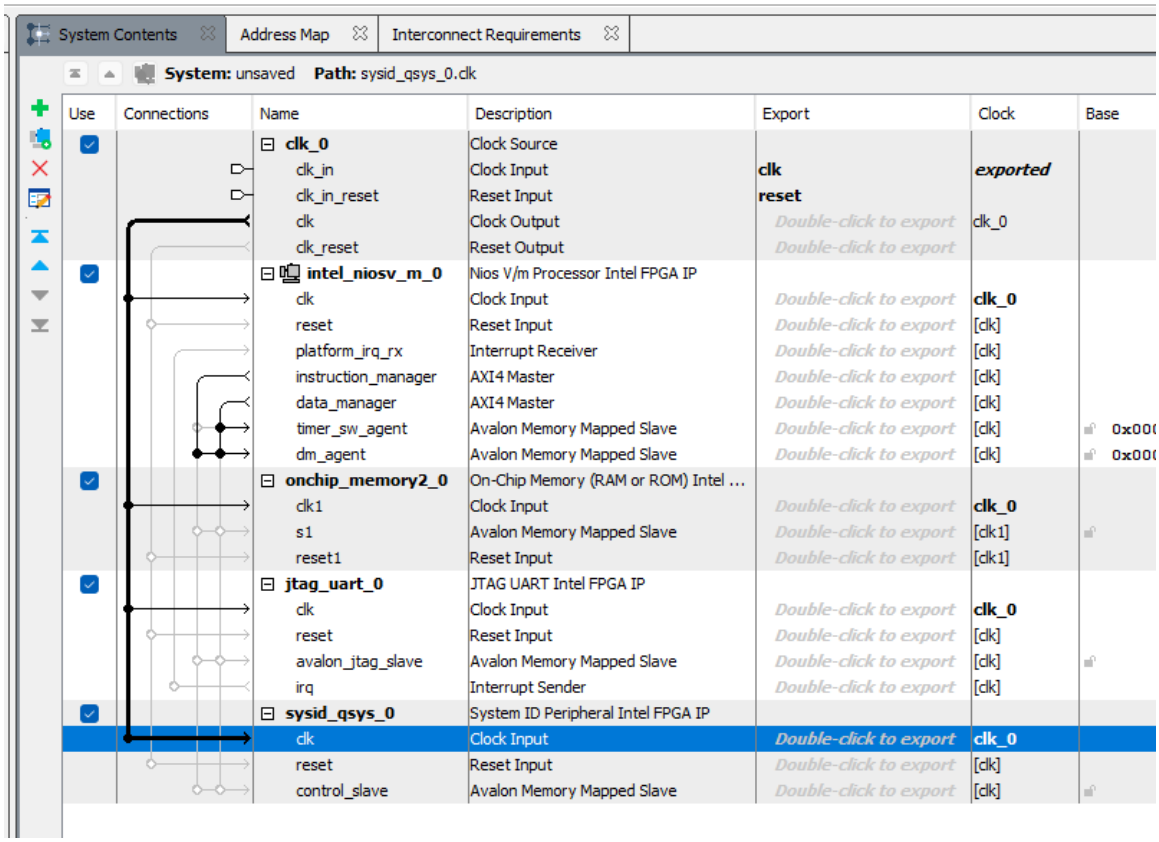


14. A configuration page will appear. There are no changes to be made. Click Finish.



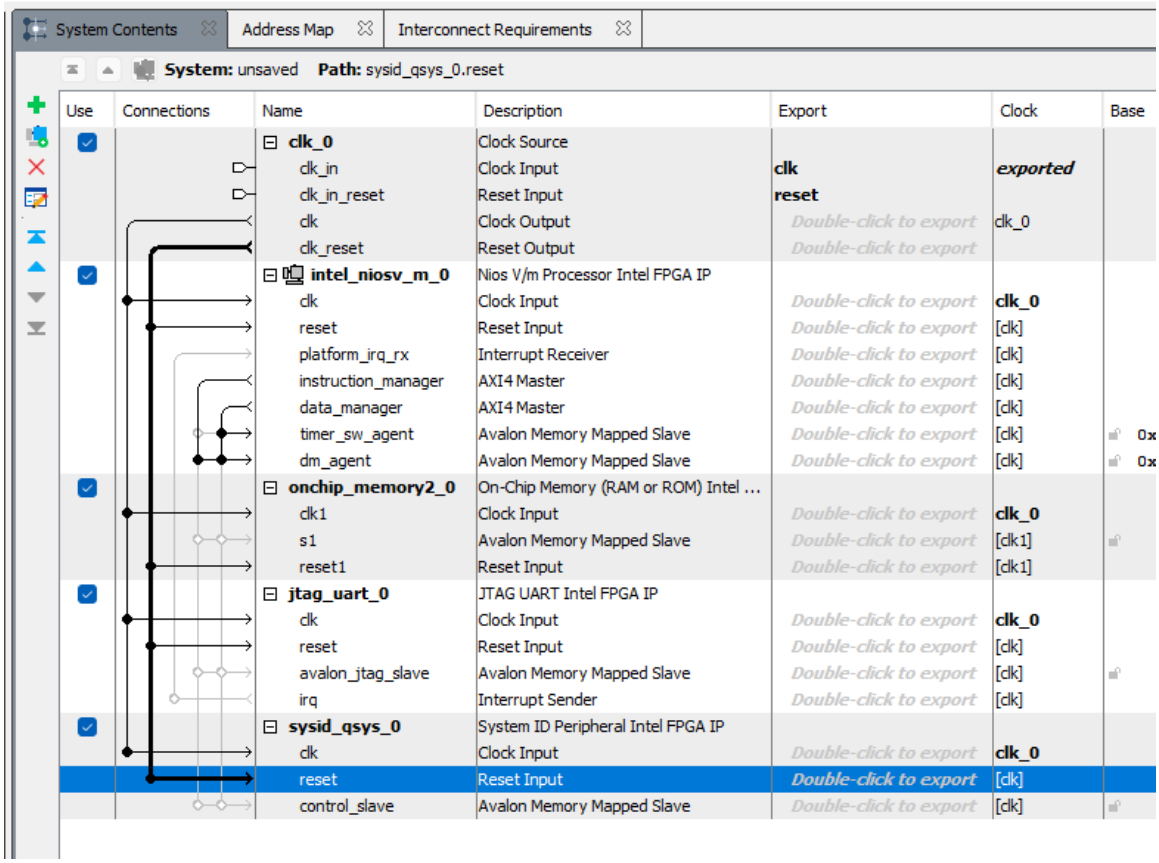
Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

15. Now, we need to wire the IP blocks together. First, wire all the clk lines together by clicking on the dots for all four IP blocks.



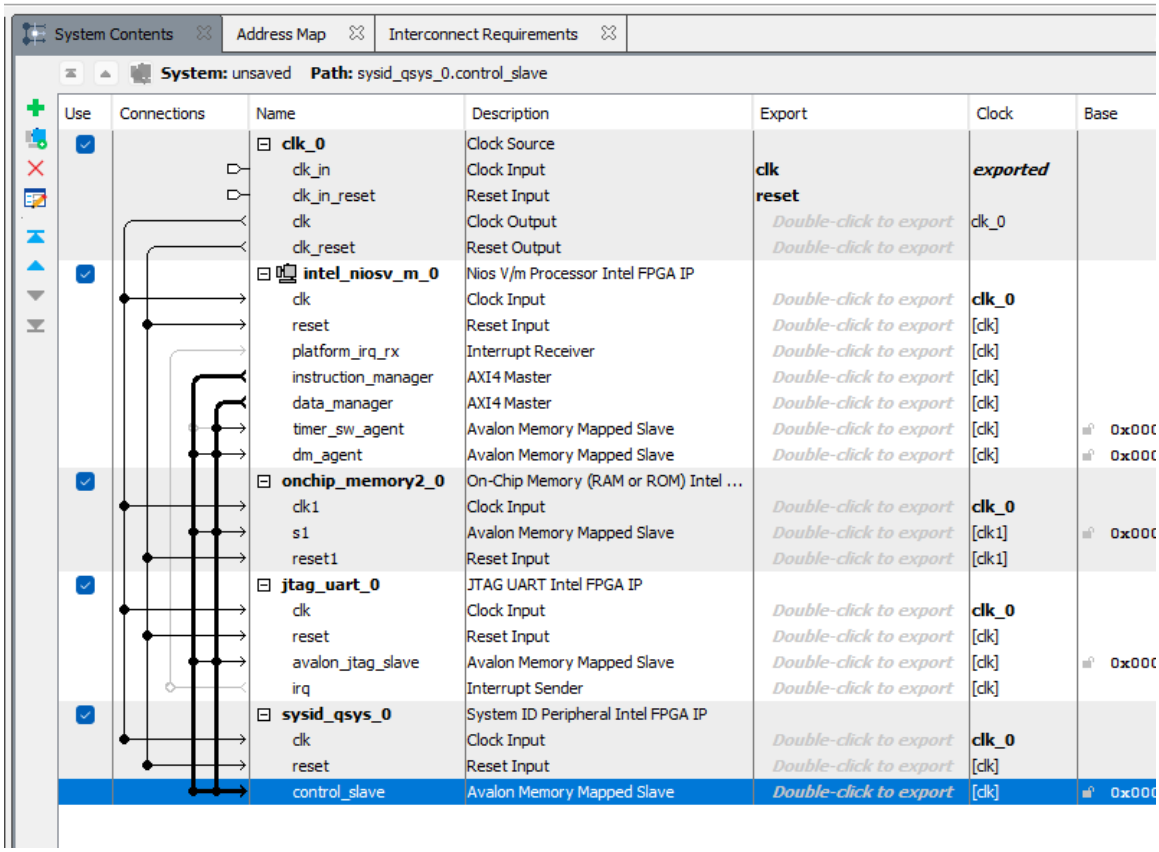
16. Next, connect all the reset lines together by clicking on the dots for all four IP blocks.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

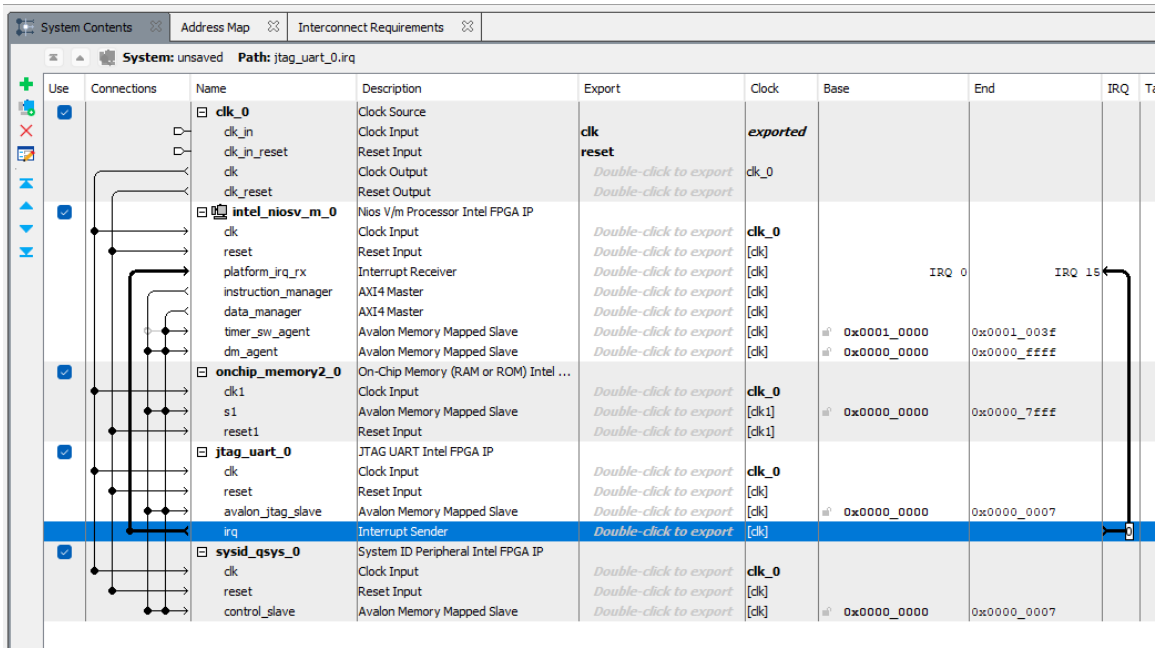


17. The memory lines have to be connected together. Connected the Instruction_manager and data_manager to all other items in the design.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

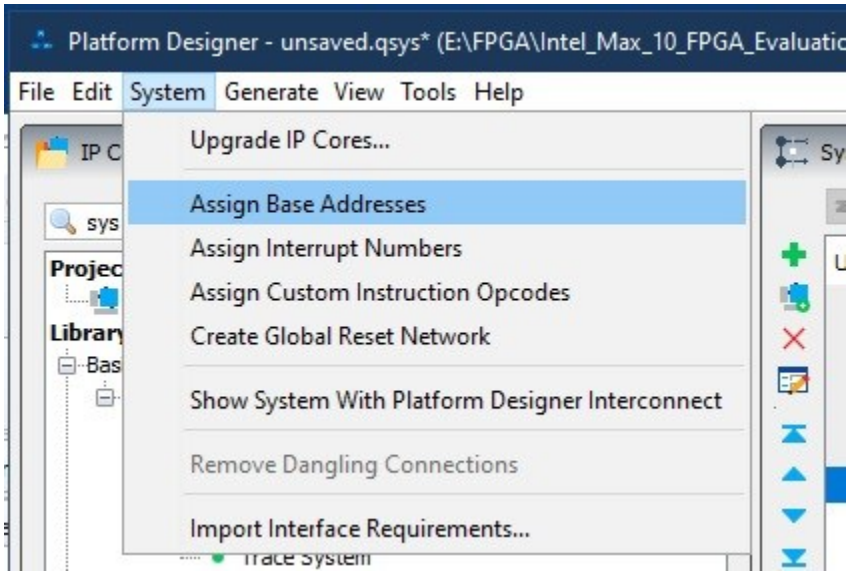


18. Finally, connect the jtaq_uart irq line to the intel_niosv_m_0.
19. If you scroll to the right, the irq is given a default value.

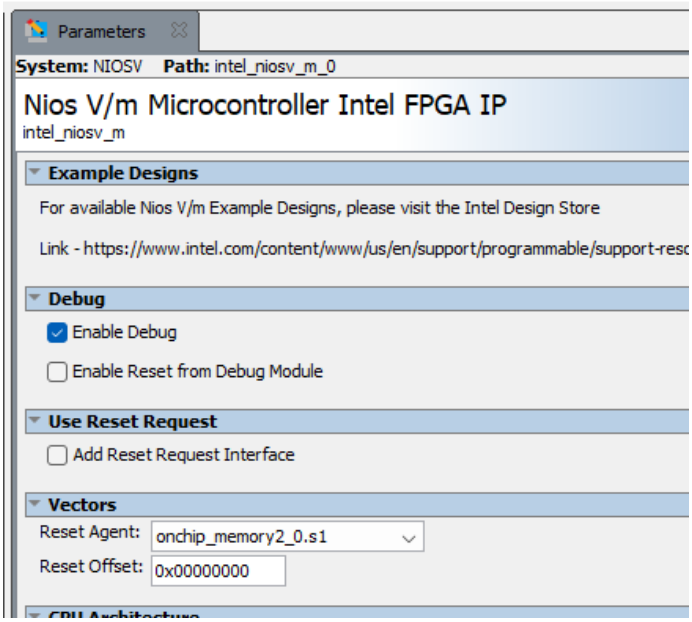


Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

20. Let's assign a base address. From the menu, select System->Assign Base Address. This will remove a number of errors from the message box.

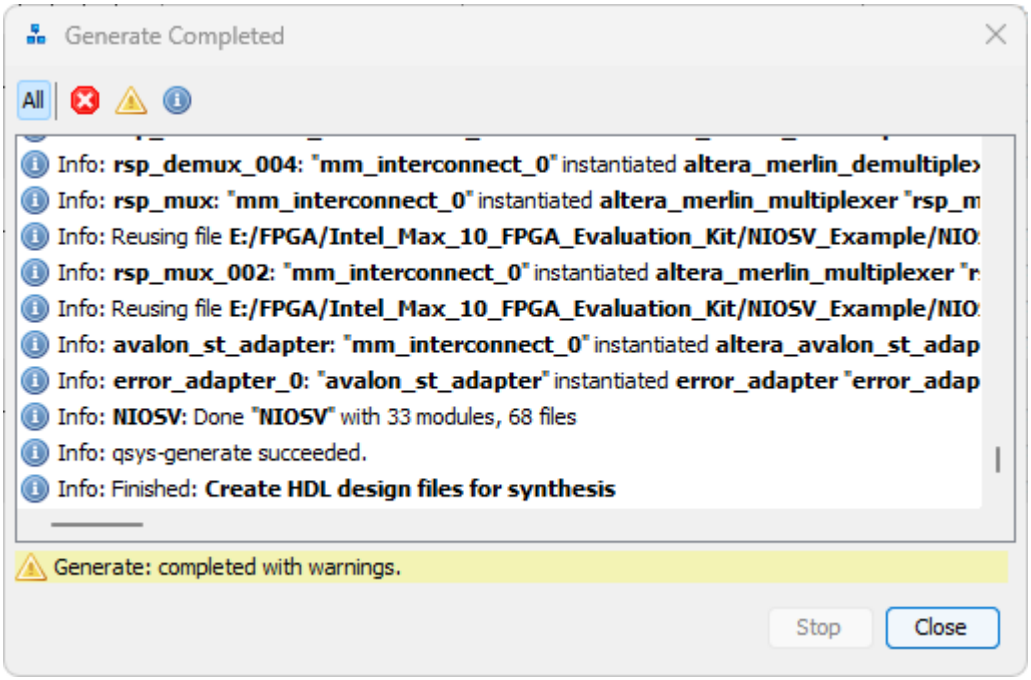


21. Finally, let's set the reset and exception vector addresses. Double-click on the intel_niosv_m_0 to open the configuration page.
22. In the Vectors section, change the Reset Agent to onchip_memory2_0.s1

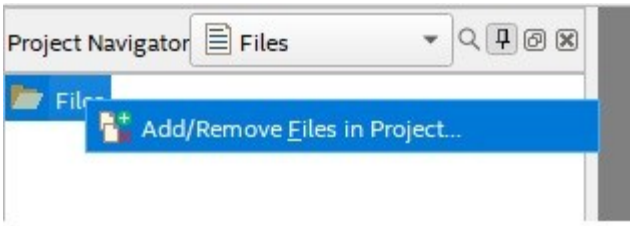


23. Click on Generate HDL…
24. A dialog appears, keep the defaults and click the Generate button.
25. A dialog will appear asking you to save the design, click Save.
26. Give the name as NIOSV.qsys and click Save.
27. Once the save has been completed, click Close.
28. The generate process kicks off. The processes should succeed with warnings, click Close.

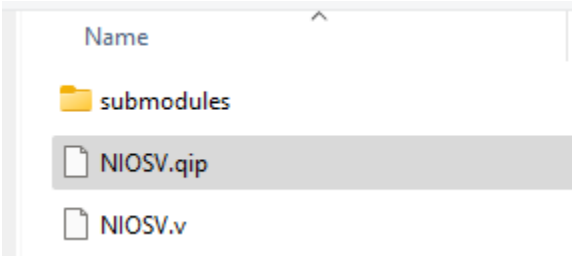
Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.



29. Click Finish to close the design.
30. Quartus then reminds you to add the new design to the project. Click Ok.
31. In the Project Navigator, click on the drop-down and select Files.
32. Right-Click on Files and select Add/Remove Files in Project.

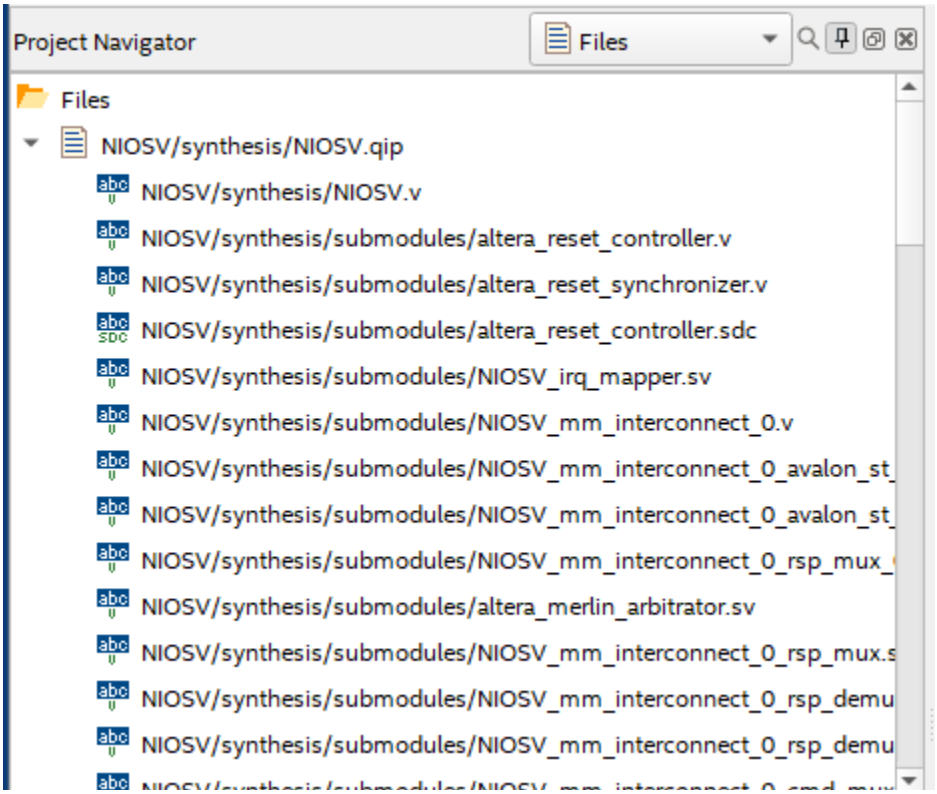


33. A Settings – NIOSVCPU page appears with Files on the left highlighted. Click the three dots, browse button for File name, and navigate to \NIOSV_Example\NIOSV\synthesis folder.
34. Click on NIOSV.qip file and click open



35. Click OK to close the Settings - NIOSVCPU page. The qip file is added to the Project navigator list. Underneath are all the Verilog files that were generated by Platform Builder.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.



36. Save the project.
37. In the project navigator, right click on NIOSV.qip and select “Set as Top-Level Entity”.
38. In the Task pane on the left, double-click on Fitter (Place & Route) to start the task. The analysis will take some time, and it should succeed in the end. This step helps to diagnose any errors and finds the Node Names for the pin assignments in the next step.

39. Once the process completes, the pin assignments need to be set. From the menu select Assignments->Pin Planner or click on the icon from the toolbar. The analysis that was just run populated the Node Name list at the bottom of the Pin Planner dialog.

Named: * Edit:

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Re
altera_reserved_tck	Input				PIN_18	2.5 V (default)	
altera_reserved_tdi	Input				PIN_19	2.5 V (default)	
altera_reserved_tdo	Output				PIN_20	2.5 V (default)	
altera_reserved_tms	Input				PIN_16	2.5 V (default)	
clk_50MHz	Input				PIN_28	2.5 V (default)	
SW1	Input				PIN_29	2.5 V (default)	
<<new node>>							

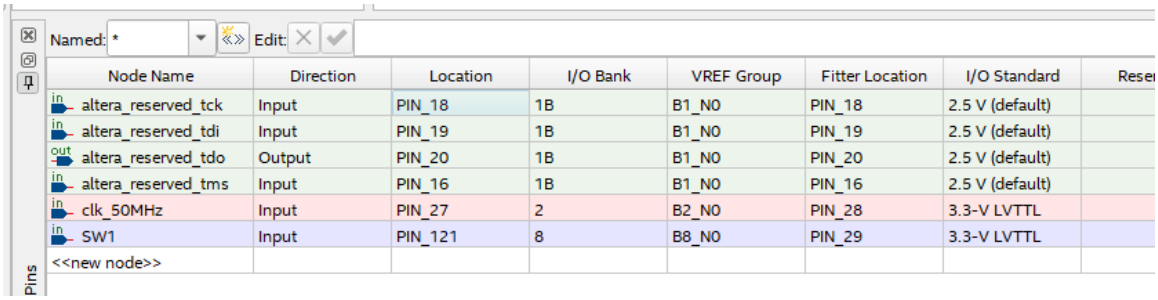
All Pins

40. Using the board schematic, locate the pins for the SW1 and the 50Mhz clock. Set the Location values for both node names. For the MAX 10 – 10M08 Evaluation Board, these values are as follows:

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

Node Name	Location
SW1	PIN_121
Clk_50MHz:	PIN_27
altera_reserved_tck	PIN_18
altera_reserved_tdi	PIN_19
altera_reserved_tdo	PIN_20
altera_reserved_tms	PIN_16

41. Set the I/O Standard to 3.3V-LVTTL for both pins. You can see from the schematic that all of the I/O are tied to 3.3V.



42. Close the Pin Planner when finished.
43. Save the project.

1.1.3 Add Design Constraints File

1. From the menu in Quartus, select File->New-> Synopsys Design Constraints File
2. Click Ok
3. Add the following:

```
#50MHz clock 1/50 = 0.02 or 20ns
#1/50000000 = 20e-9 20ns
#create_clock -period "50Mhz" -name clk_50MHz clk_clk

create_clock -period 20.000 -name clk_50MHz clk_clk

derive_clock_uncertainty

set_false_path -from [get_ports reset_reset_n]
set_false_path -from [get_ports altera_reserved*]
set_false_path -from * -to [get_ports altera_reserved_tdo]
```

4. Save the file as NIOSVCPU.sdc

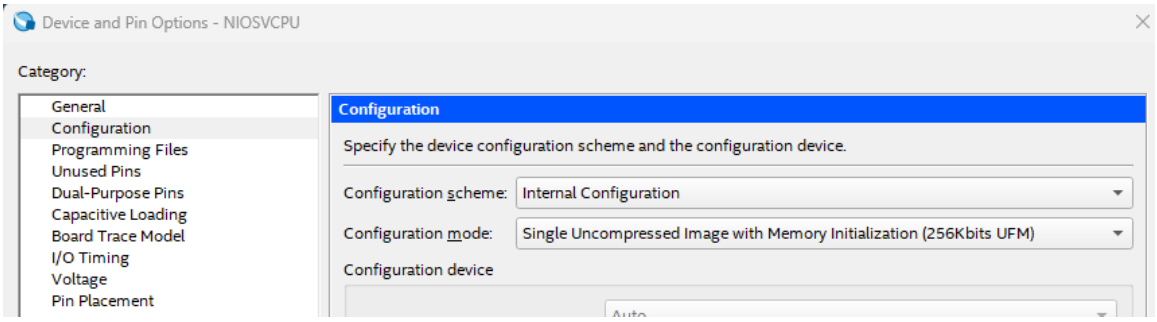
1.1.4 For MAX 10, set configuration scheme

Set the configuration for the device.

1. From the Quartus menu, select Assignments->Device
2. Click on the “Device and Pin Options…” button.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

3. Set the internal configuration mode to “Single uncompressed image with Memory initialization (256Kbits UFM)”

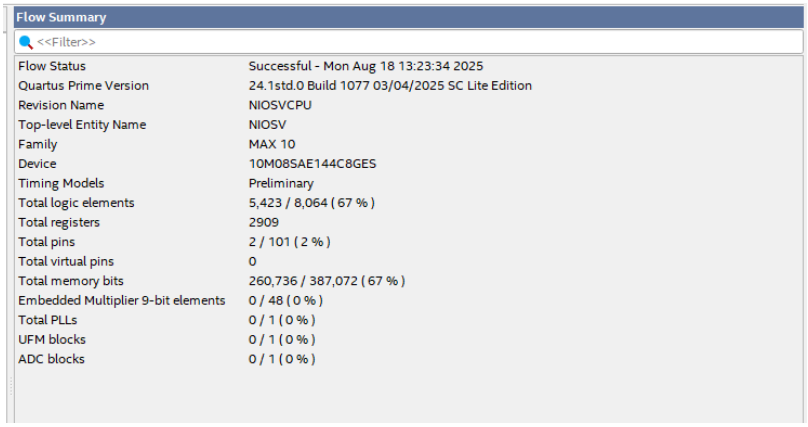


4. Click OK
5. Click OK

1.1.5 Compile the Project

Note: A best practice at this point would be to make a backup of the project folder. Quartus can crash unexpectedly.

1. Finally, compile the design. In the Task pane, right-click on Compile and Design and select Start from the context menu, or you can click on the symbol in the toolbar. The compile should complete successfully.



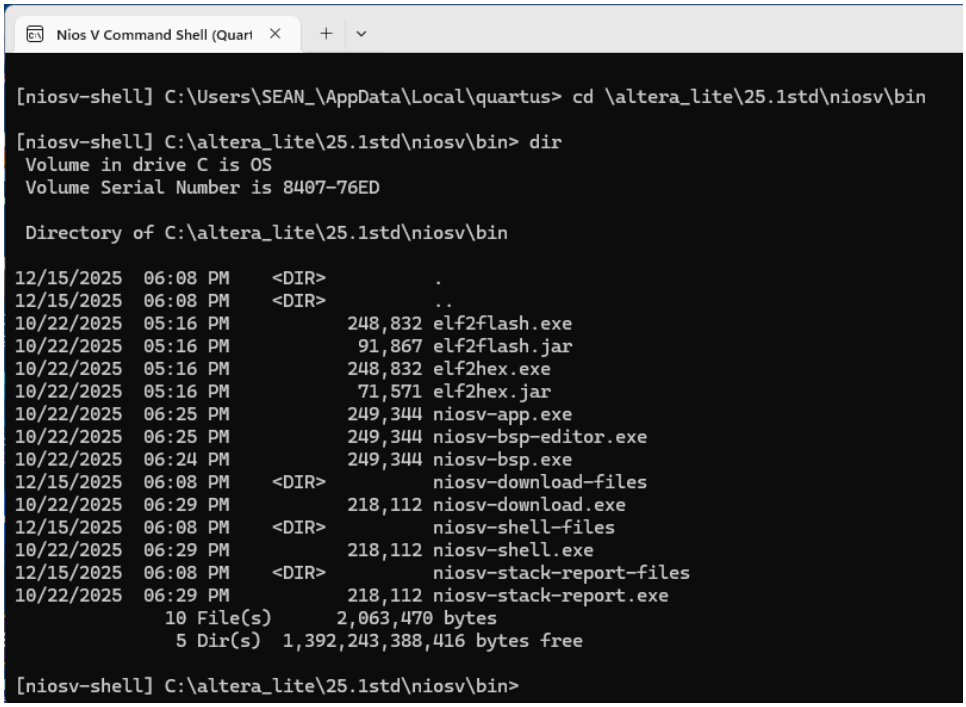
1.2 Part 2: Nios V Shell Create BSP and Cmake Application

For Nios II, the next step would be to open Eclipes and create the software project with the .sopcinfo file, which also creates the BSP. The Nios V application development process is not as integrated as the more mature Nios II. There are a few more manual steps before we get to the IDE environment. The good news is that Nios V and RiscFree IDE do not require WSL if you are running on Windows. Only Cmake is required. This section will create the BSP and basic application project, and the following section will be used to create the application in RiscFree IDE.

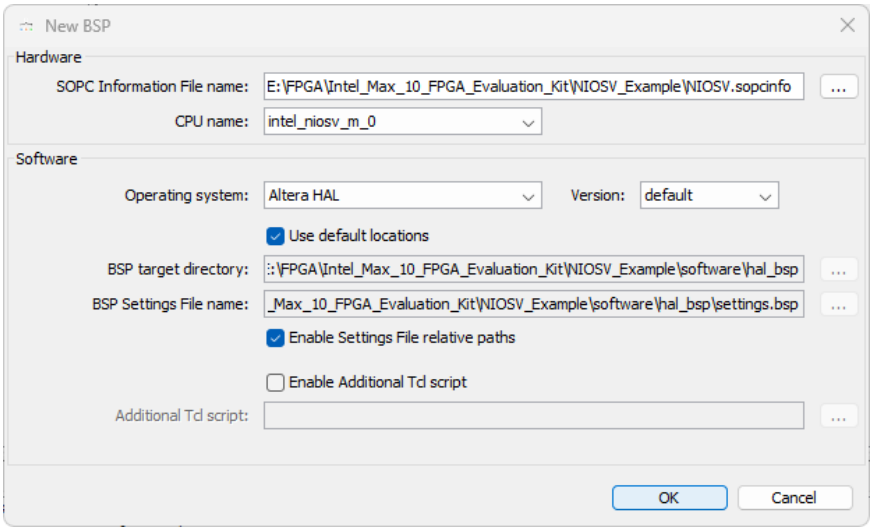
1. From the Start menu, open the Nios V Command Shell.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

2. A command window appears. Change directory to C:\altera_lite\25.1std\niosv\bin> if the window doesn't open to the folder.
3. Enter DIR and hit enter. You will see a number of applications available.

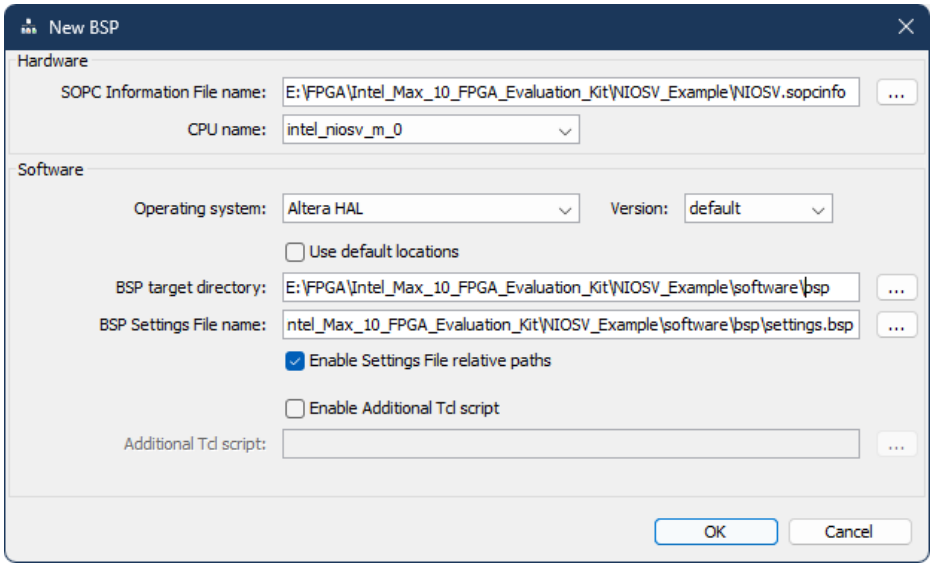


4. Run niosv-bsp-editor.exe
5. The BSP Editor opens up.
6. From the menu, select File-New BSP…
7. Click on the button with 3 dots for "SOPC Information File name:"
8. Open the NIOSVCPU.sopcinfo. As the file opens, the file information is read and fills the rest of the dialog.



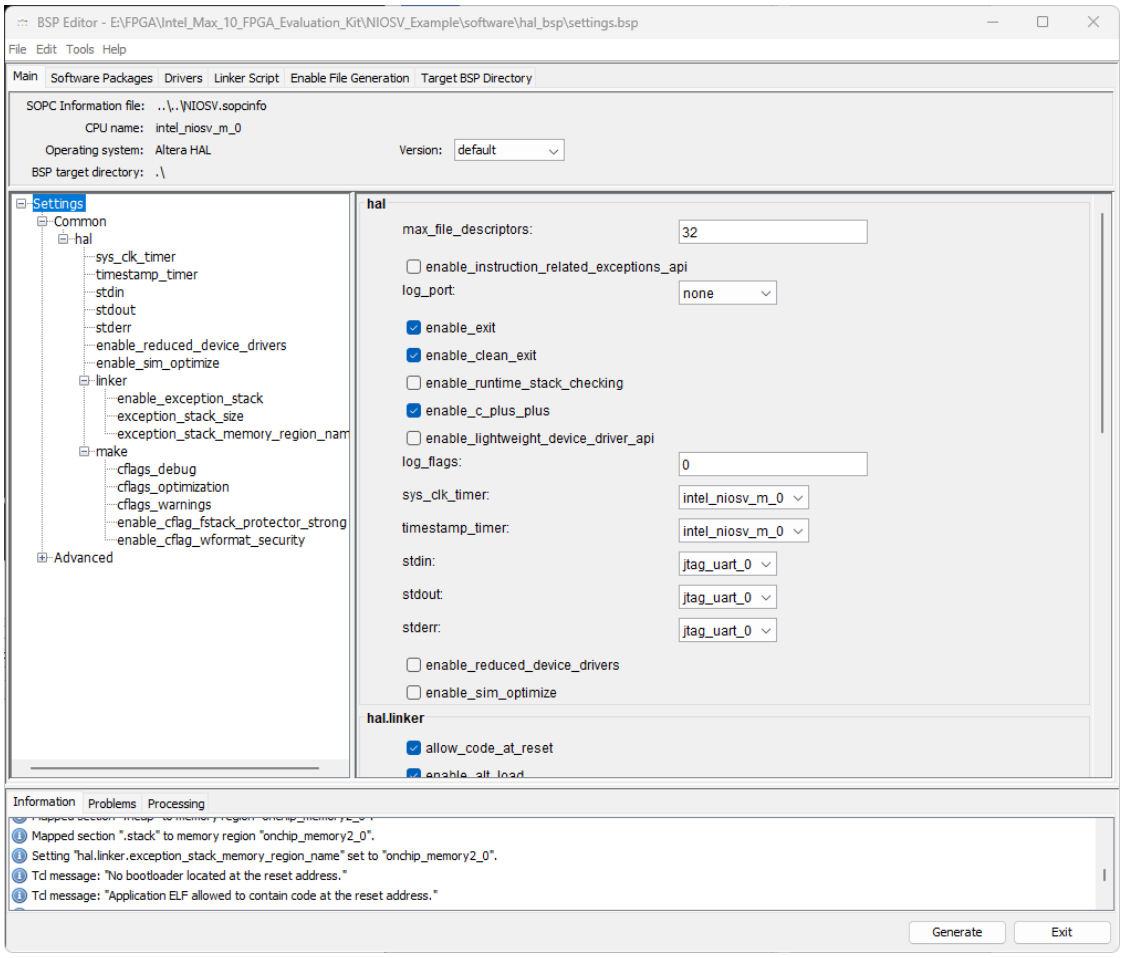
Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

9. Uncheck the Use default locations and change the BSP target directory from hal_bsp to bsp.The BSP will be created in a software\bsp subfolder under the \NIOSV_EXAMPLE project.



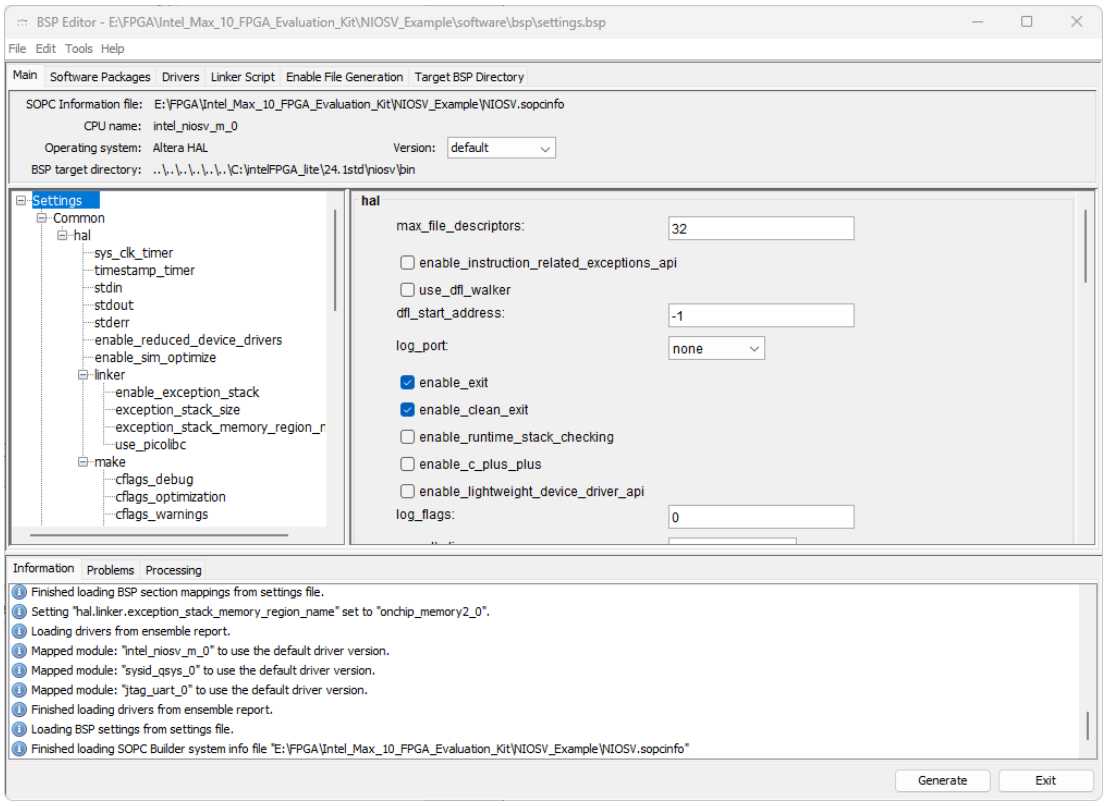
10. Just like Nios II, the BSP editor provides all the configurable items available in the BSP. Click OK.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

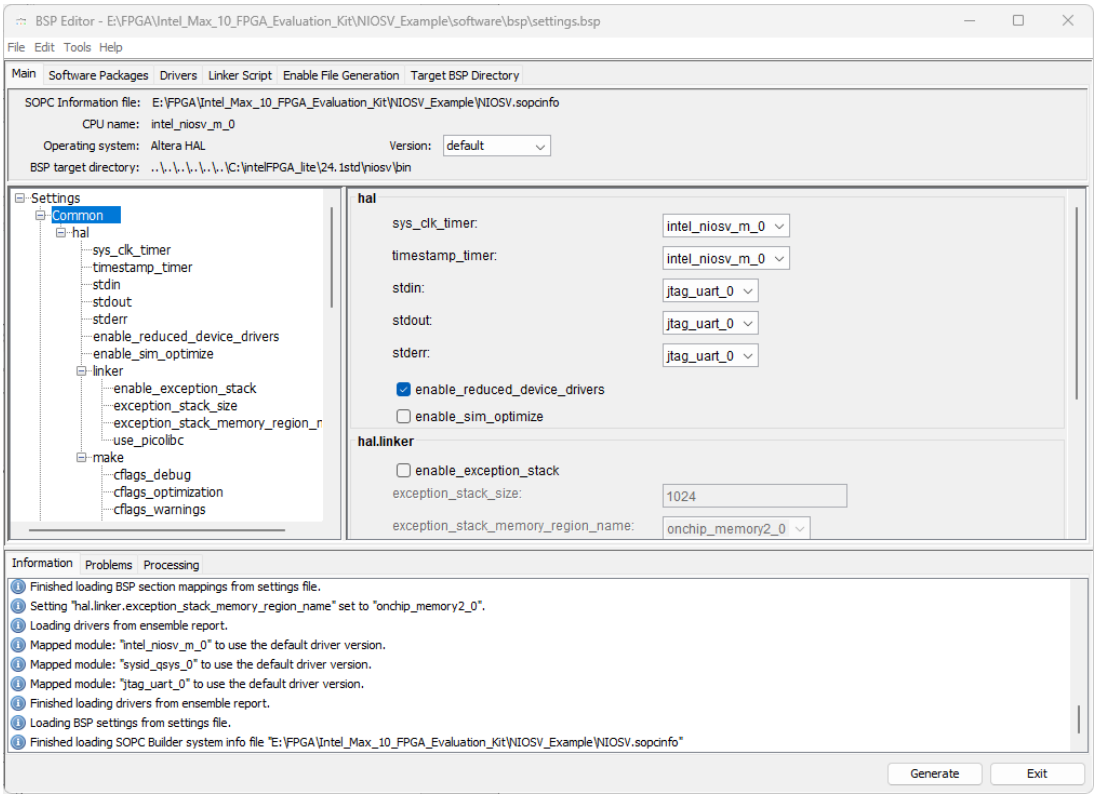


11. Under Settings root, uncheck “enable_c_plus_plus”

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

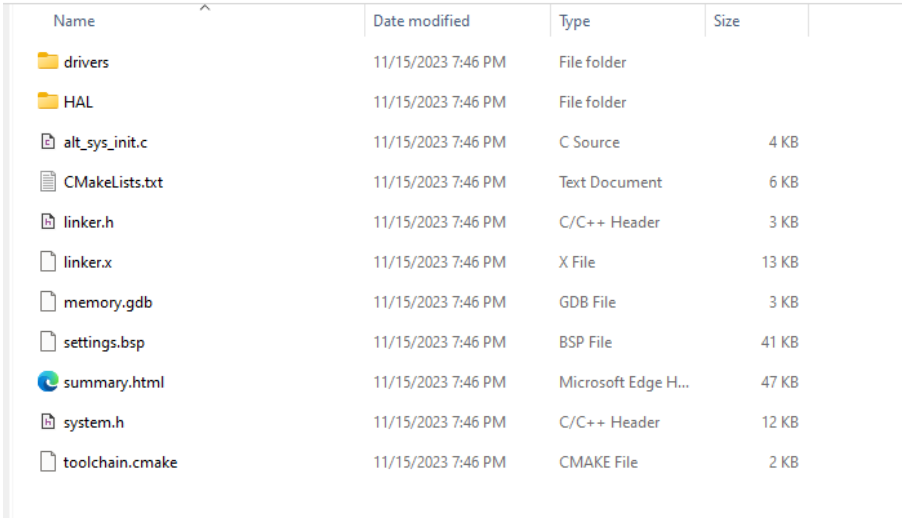


12. Under Common, check enable_reduced_device_drivers

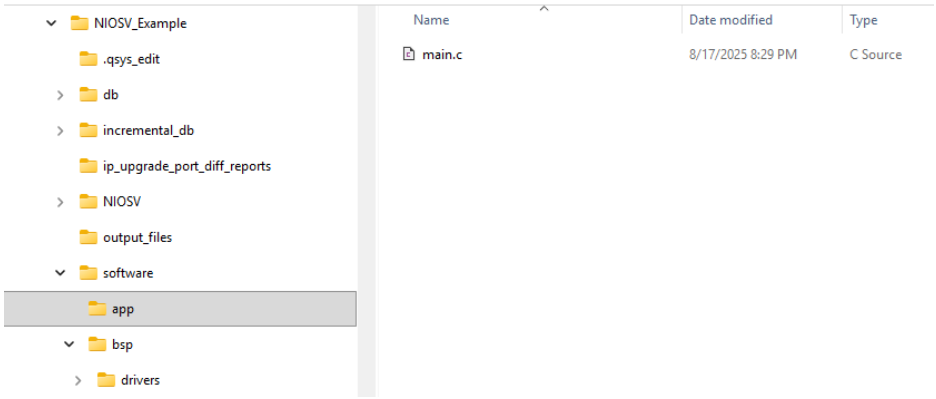


Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

13. Click the Generate button.
14. Once the BSP has been generated, click on Exit to close the BSP Editor.
15. Open File Explorer, and navigate to \NIOSV_Example\software folder. You will see what BSP files were generated.



16. Under \NIOSV_Example\software folder, create a subfolder called “app.”
17. Create a text file called main.c in the \NIOSV_Example\software\app folder.



18. In the Nios V command shell, change directory to the \NIOSV_Example\software\app folder.
19. Run the following command:

```
niosv-app -b=../bsp -a=. -s=main.c
```

20. A CmakeList.txt file gets generated, open the text file in an editor.

cmake_minimum_required(VERSION 3.14)
add_subdirectory(../bsp bsp)
include(../bsp/toolchain.cmake)

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

```
project(app)

enable_language(ASM)
enable_language(C)
enable_language(CXX)

add_executable(app.elf)

target_sources(app.elf
    PRIVATE
        main.c
)

target_include_directories(app.elf
    PRIVATE
    PUBLIC
)

target_link_libraries(app.elf
    PRIVATE
        -T "${BspLinkerScript}" -nostdlib
        "${ExtraArchiveLibraries}"
        -Wl,--start-group "${BspLibraryName}" -lc -lstdc++ -lgcc -lm -Wl,--end-group
)

# Create objdump from ELF.
set(objdump app.elf.objdump)
add_custom_command(
    OUTPUT "${objdump}"
    DEPENDS app.elf
    COMMAND "${ToolchainObjdump}" "${ToolchainObjdumpFlags}" app.elf >
            "${objdump}"
    COMMENT "Creating ${objdump}."
    VERBATIM
)
add_custom_target(create-objdump ALL DEPENDS "${objdump}")

# Report space free for stack + heap. Note that the file below is never created
# so the report is always output on build.
set(stack_report_file app.elf.stack_report)
add_custom_command(
    OUTPUT "${stack_report_file}"
    DEPENDS app.elf
    COMMAND niosv-stack-report -p "${ToolchainPrefix}" app.elf
    COMMENT "Reporting memory available for stack + heap in app.elf."
    VERBATIM
)
add_custom_target(niosv-stack-report ALL DEPENDS "${stack_report_file}")

# Generate HEX file(s) from app.elf using elf2hex tool.
# Note : If ECC Full is enabled, width of 39 is set for NiosV TCM. Otherwise, 32.
add_custom_command(
    OUTPUT "onchip_memory2_0.hex"
```

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

```
    DEPENDS app.elf
    COMMAND elf2hex app.elf -o onchip_memory2_0.hex -b 0x00010000 -w 32 -e 0x00017CFF -r 4
    COMMENT "Creating onchip_memory2_0.hex."
    VERBATIM
)
add_custom_target(create-hex ALL DEPENDS "onchip_memory2_0.hex")
```

1.3 Part 3: Write the helloapp Application in RiscFree IDE for Intel FPGAs

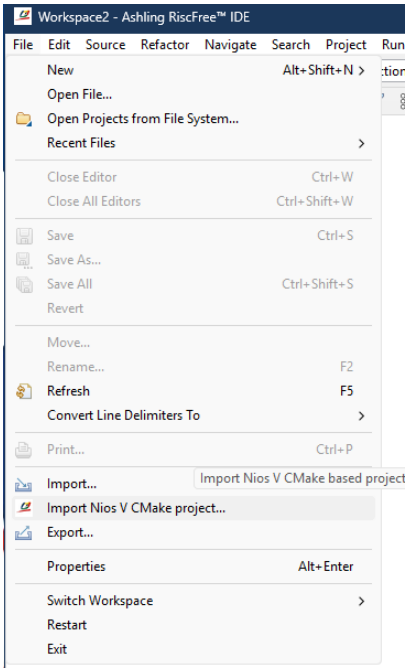
Now, we are ready to write the applications in the RiscFree IDE.

1. Open niosv-shell if not already open.
2. Type the following and hit enter to start RiscFree IDE:

   riscfree.exe

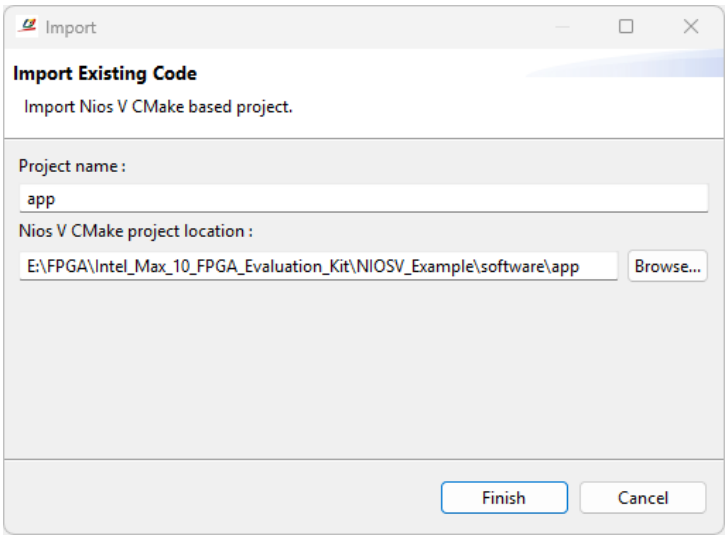
Warning: The reason to open RiscFree.exe via niosv-shell is to take advantage of the niosv shell path settings so the build tools can run the HEX generation convertions. If RiscFree.exe is run outside the niosv-shell environment, the application will be built, but fail on the HEX conversion.

3. Keep the default work space and click Launch.
4. Click on Create Project or from the menu select File->"Import Nios V CMake project…"

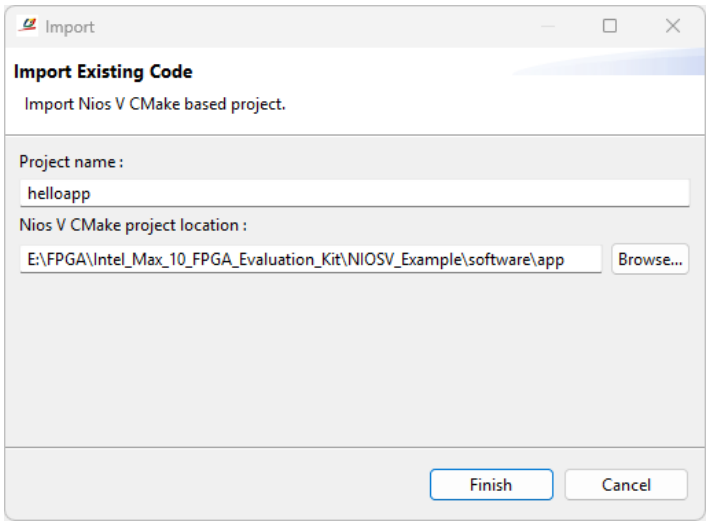


5. In the import dialog, click on browser and open the app folder location. The project name will automatically fill with the folder name.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.



6. You can change the name if you wish. For this example, the project name will be helloapp



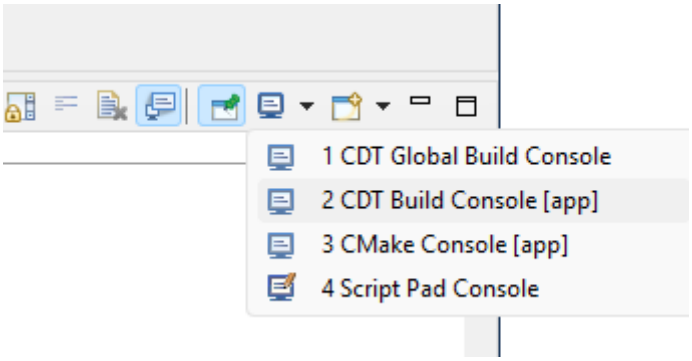
7. Click Finish.
8. Open the main.c and fill in the following code:

```
#include "sys/alt_stdio.h"

int main(){

        alt_putstr("Hello from Nios V!\n");
        alt_putstr("Programming for RISC V is fun!\n");

        int x = 5;
        int y = 10;
        int z = x * y;

        //Never exit the program
        while(1);
```

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

```
        return 0;

}
```

9. Near the bottom right, set the console output to CDT Build Console [app].



10. Right-click on app project and select Build Project from the context menu. The build should complete successfully with a app.elf file being created.

```
Problems  Tasks  Console  Properties  Search
CDT Global Build Console
[ 97%] Creating app.elf.objdump.
[ 98%] Reporting memory available for stack + heap in app.elf.
[100%] Creating onchip_memory2_0.hex.
TBBmalloc: skip allocation functions replacement in ucrtbase.dll: unknown prologue for function _msize
[100%] Built target create-objdump
app.elf
 * 26.49 KB - Program size (code + initialized data).
 * 3020 B - Free for stack + heap.
[100%] Built target create-hex
[100%] Built target niosv-stack-report

19:21:27 Build Finished. 0 errors, 1 warnings. (took 14s.5ms)
```

Notice that the program size is 26.49 KB and free space for the stack and heap is only 3020 Bytes. For NiosII, a similar program was only 732 bytes in size so there is a big increase in memory required.

Warning: If you open RiscFree IDE independently of the niosv-shell, an error will appear at the end of the build. The elf file was built, but the paths to run the stack report and create HEX files were invalid. You can always run the niosv-stack-report.exe and elf2hex.exe seperatly.

```
Problems  Tasks  Console  Properties  Search
CDT Global Build Console
[ 98%] Creating app.elf.objdump.
[100%] Reporting memory available for stack + heap in app.elf.
process_begin: CreateProcess(NULL, niosv-stack-report -p riscv32-unknown-elf- app.elf, ...) failed.
make (e=2): The system cannot find the file specified.
make[2]: *** [CMakeFiles/niosv-stack-report.dir/build.make:73: app.elf.stack_report] Error 2
make[1]: *** [CMakeFiles/Makefile2:177: CMakeFiles/niosv-stack-report.dir/all] Error 2
make[1]: *** Waiting for unfinished jobs....
[100%] Built target create-objdump
make: *** [Makefile:91: all] Error 2
"C:/intelFPGA_lite/24.1std/riscfree/build_tools/bin/make.exe -j all" terminated with exit code 2. Build might be incomplete.

14:01:01 Build Failed. 4 errors, 1 warnings. (took 13s.246ms)
```

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

1.4 Part 4: Test the Design and the Application

With the design and application compiled, we can now test the design and application on the board.

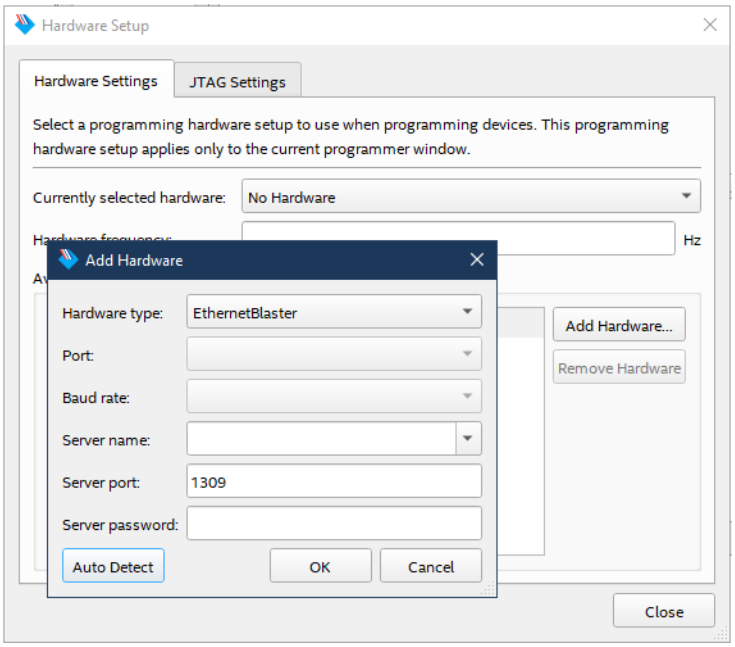
1.4.1 Program the Board

We will go back to Quartus and program the board with the design.

1. Connect the board and the programming cable together per the cable instructions.

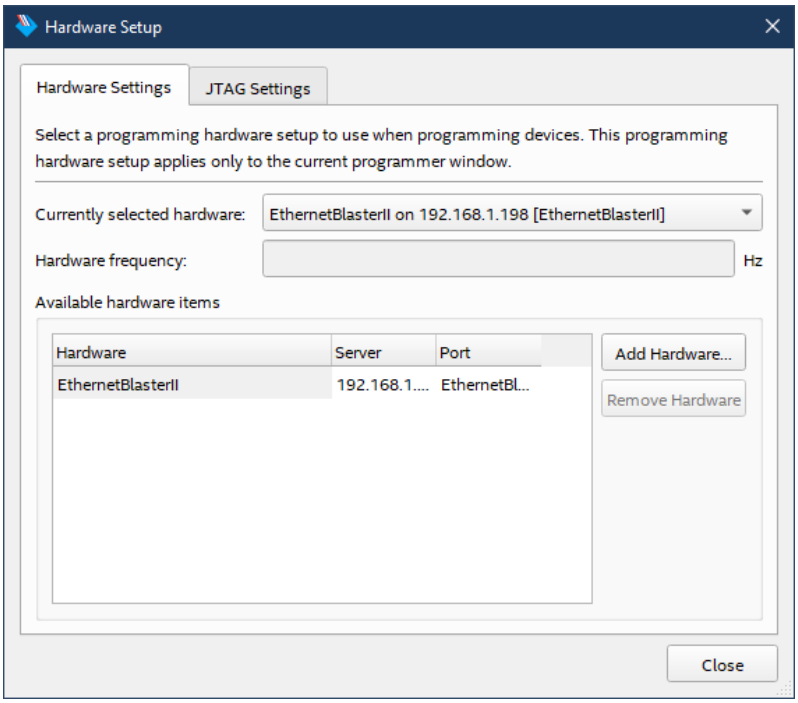
Note: The MAX 10 – 10M08 Evaluation Kit doesn't come with a programming cable or built-in JTAG USB Blaster II. You will have to use either the USB Blaster II or EthernetBlaster II external cables. The EthernetBlaster II was used. DHCP setup was not working, so a direct Ethernet cable connection was made between a PC and the EthernetBlaster II. Set the static IP for the PC network card to 198.162.0.1. Access the EthernetBlaster II via a browser and then change the IP to a static IP that matches the network. The new IP address was used as the Server name. Your experience might be different.

2. Power on the board and the programming cable box.
3. In Quartus Prime, from the Task pane, right-click on Program Device (Open Programmer) and select Open from the context menu or click on the icon on the toolbar.
4. The Programmer dialog appears, click on the "Hardware Setup" button.
5. Click the Add hardware button, select the Hardware type, and fill in any remaining information, and click OK.

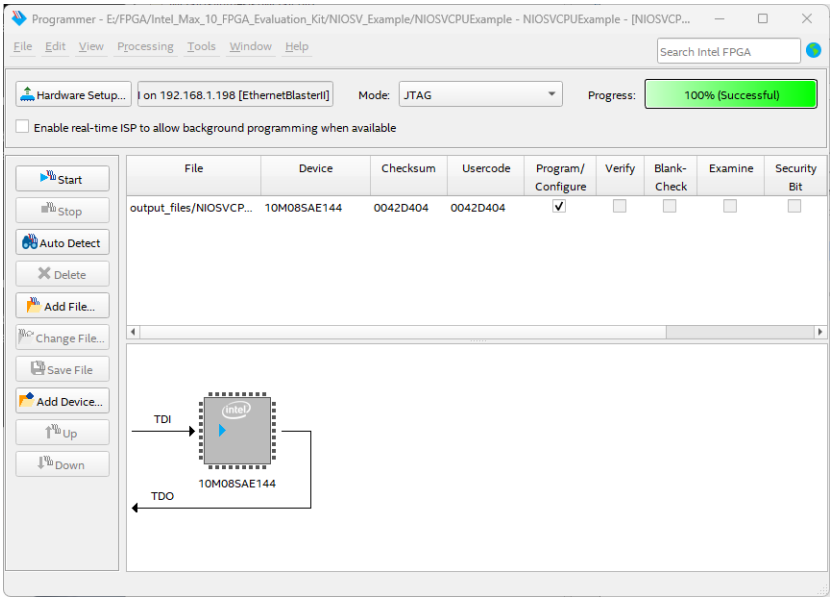


6. The tool allows you to connect to a number of programming cables. We need to select the one for our board. In the "Currently selected hardware", click the drop-down, select the hardware cable for the board, and click Close when finished

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.



7. An NIOSVCPUExample2.sof file gets created during the Compile Design flow. The file is automatically filled in. There is only one FPGA on the board and in the JTAG chain so the file already has the Program/Configure checkbox checked. Click the Start button to program the board. The process takes a few seconds and shows that the task was completed successfully.



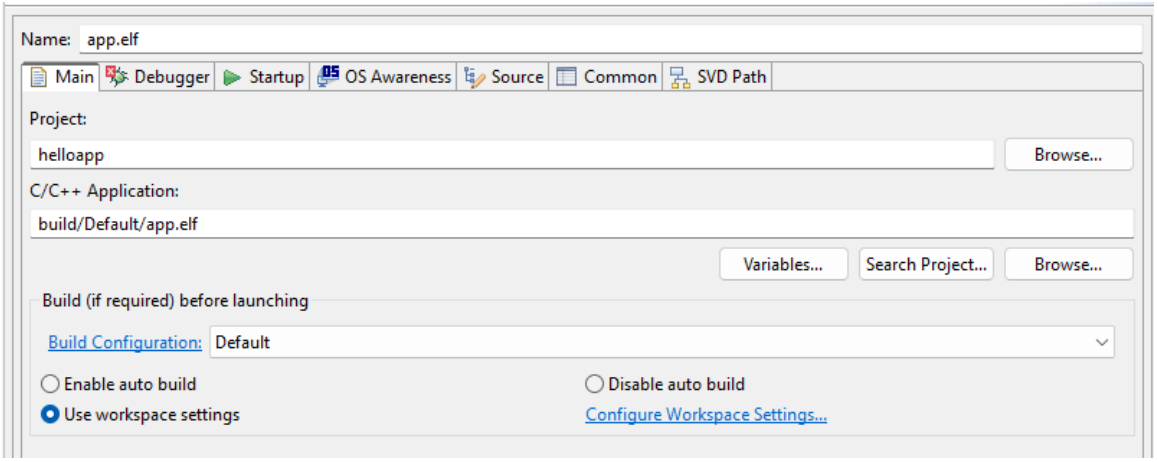
Notice that there is no time constraint dialog box popping up. The Nios V is free and doesn't required a paid license.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

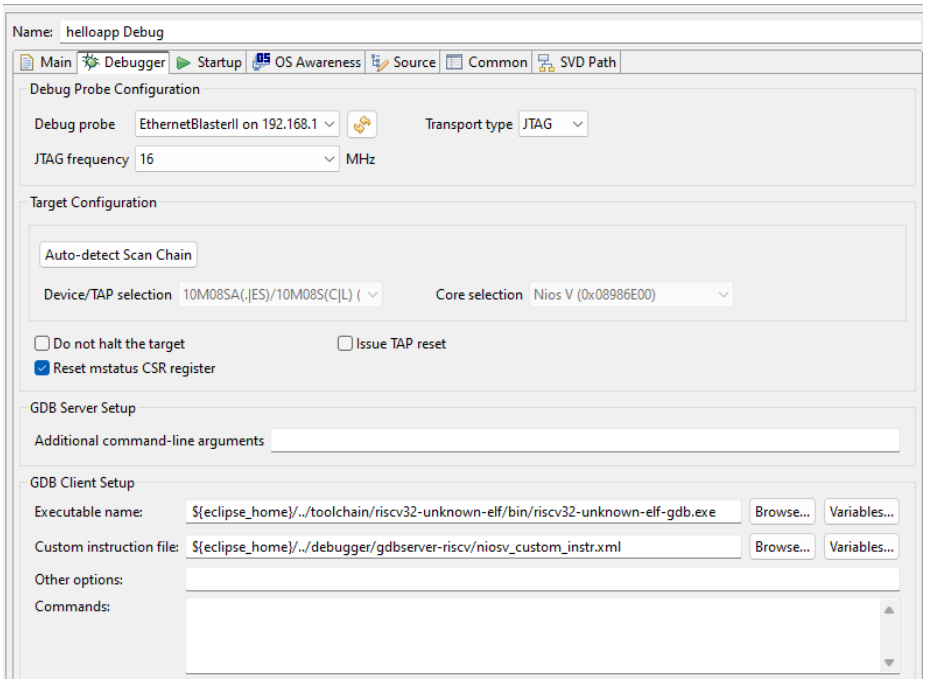
1.4.2 Debug the Application

Now, we go back to the RiscFree IDE to download and debug the applicaiton

1. With the JTAG cable connected, in RiscFree IDE, right click on the helloapp in Project Explorer.
2. Select Debug AS->Debug Configurations…
3. Click on Ashling RISC-V hardware Debugging
4. In the Main tab, you should see the project pointing to the Project and the app.elf file.



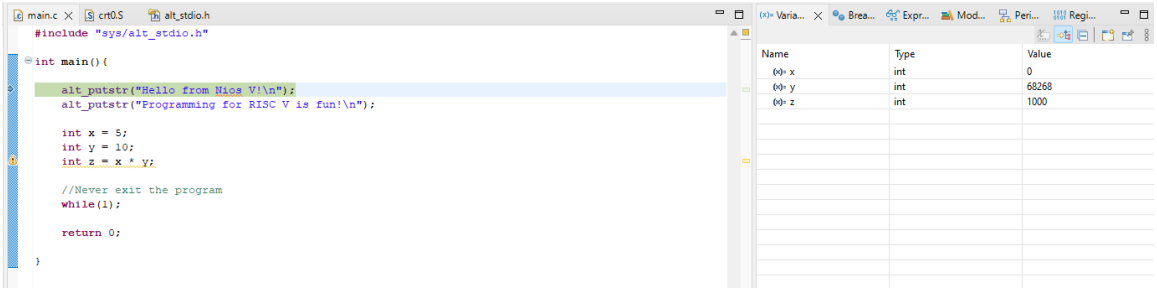
5. Click on Debugger tab.
6. Set the Debug probe to the debug probe connected to the board.
7. Set the JTAG frequency to 16 MHz
8. Click on the “Auto-detect Scan Chain”. The Devcie/TAP selection should be filed in with the 10M08 FPGA and the Core selection should be the Nios V.



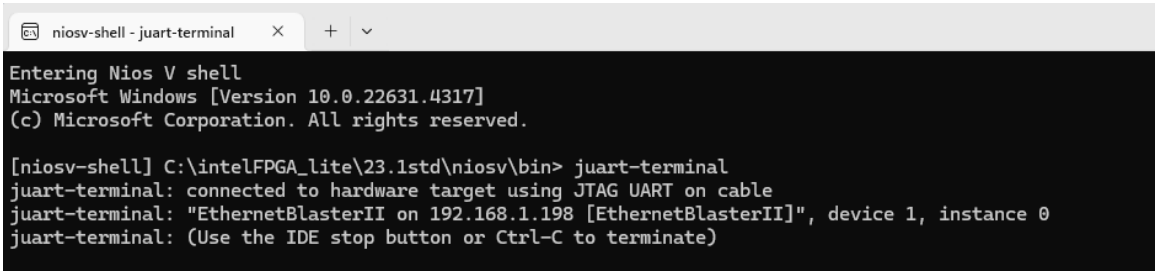
9. Click Apply

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

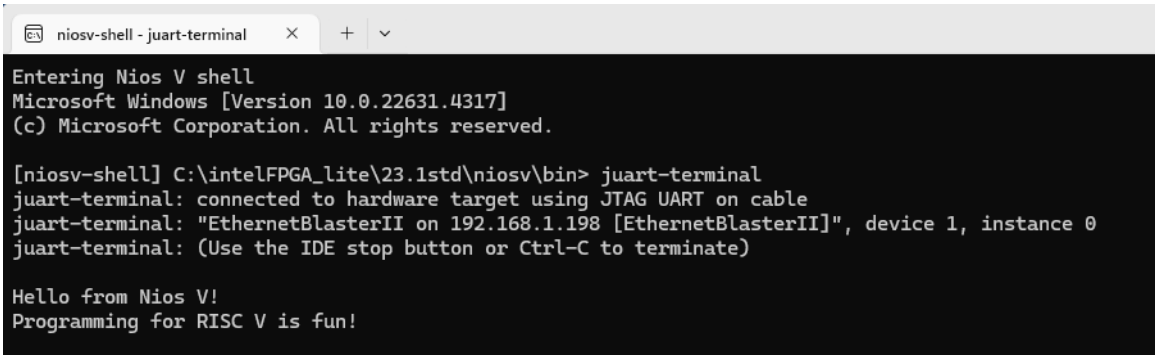
10. Click Debug. The application will download and stop at the first line in Main().



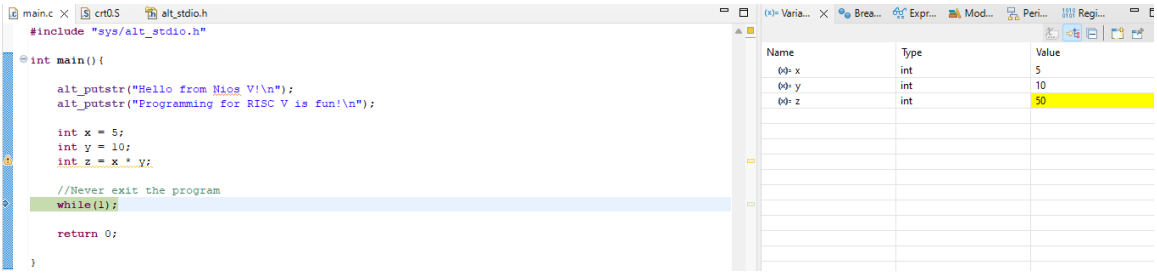
11. Open NIOSV-Shell
12. Run the following command to start the terminal application: `juart-terminal`



13. In RiscFree IDE, step throught the code. You will see the messages sent to the juart-terminal.



14. Like any debugger, as you continue to step through the code you can see the variables changes, as well as other processor registers and memory.



15. Stop debugging when fnished.

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.

1.5 Nios V Applications are Bigger, a New Platform Moving Forward

As called out in the application build results, the Nios V applications are bigger, which is too big for the MAX10 10M08 FPGA. If there was external RAM memory available, then the program would fit easily. Intel® MAX® 10-10M08 Evaluation Kit has been a fun cheap board to work with, but the platform is not the best to continue with Nios V examples. Future articles and other publications will be using the MAX10 10M50 board with RAM and ROM.

1.6 Summary: A Change in the Process

The article covered the basic steps to create a FPGA design for the Nios V and then create and debug an application using the Ashling RiscFree™ IDE. Since Nios V is free there is no limitation like teathering or timeout like there is with the Nios II. The development process to write Nios V applications is very different than the Nios II, but in a couple artciles, we will see how VS Code simplifies the process.

1.7 References

The following references were used for this article:

RiscFree IDE Documentation:
https://www.intel.com/content/www/us/en/docs/programmable/730783/24-2/about-the-ide.html

Video: Debugging the Nios® V Processor Using the RiscFree* IDE for Intel® FPGAs
Video: Build A Soft Core CPU - Part Three - NIOS II in Intel FPGA

Copyright © 2025 Annabooks, LLC. All rights reserved
Intel is a registered trademarks of Intel Corporation
Quartus, Nios II, Nios V, and MAX 10 are registered trademarks of Altera
All other copyrighted, registered, and trademarked material remains the property of the respective owners.