1 Tutorial 1: Hello World on the Nios II platform, Cyclone III FPGA



Overview

This tutorial will demonstrate how to generate hardware and related software interfaces in the form of RTL (Register Transfer Logic) HDL descriptions and software system libraries appropriate for use with the **Altera Quartus II**, **SOPC Builder**, and **Nios II IDE** software tools.

The sample project will implement a trivial **Hello World** application. This sample project includes a software process (running on the Nios II processor) and a hardware process (running in the FPGA) that communicate via a single stream over the **Avalon** bus. Onchip memory inside the FPGA is sufficient for accommodating the **Hello World** executable file.

The purpose of this tutorial is to take you through the entire process of generating hardware and software interfaces and importing the relevant files to the Altera environment. The tutorial will also describe how to create the platform and downloadable FPGA bitmap using the **Altera** tools. Finally, you will learn how to build and run an Impulse C software application on the **Nios II** processor using the **Nios II IDE**.

The hardware platform used in this tutorial is Altera Cyclone III Evaluation Kit, featuring Altera Cyclone III EP3C25 FPGA, and a touch-screen LCD display.

This tutorial will require approximately 90 minutes to complete, including software run times.

Steps

Loading the Hello World Application Compiling the Application for Simulation Building the Application for the Target Platform Exporting Files from CoDeveloper Creating a Quartus Project Creating the New Platform Configuring the New Platform Generating the System Generating the FPGA Bitmap Running the Application on the Platform

Note: This tutorial assumes you have purchased or are evaluating the **CoDeveloper Platform Support Package** for **Altera Nios II**, and that you have installed and have valid licenses for the **Altera Quartus II**, **SOPC Builder**, and **Nios II IDE** products.

1

1.1 Loading the Hello World Application

Hello World Tutorial for Nios II, Step 1

To begin, start the CoDeveloper Application Manager by selecting Application Manager from the Start - > Programs -> Impulse Accelerated Technologies -> CoDeveloper program group.

Note: this tutorial assumes that you have already read and understand the basic "Hello World" tutorial presented in the CoDeveloper User's Guide.

Open the Altera Nios II Hello World sample project by selecting Open Project from the File menu, or by clicking the Open Project toolbar button. Navigate to the .\Examples\Embedded\HelloWorld_NIOS directory within your CoDeveloper installation. (You may wish to copy this example to an alternate directory before beginning.) Opening the project will display a window similar to the following:

<pre>Ele Edt Yew Project Iools Window Help File Edt Yew Project Iools Window Help Froject Exchange Froject Files Frojec</pre>	- Impulse CoDeveloper Application Man	nager Universal Edition - [HelloWorld] - [HelloWorld_hw.c]
Image: Solution HelloWorld_sw.c Image: Solution HelloWorld_sw.c <td>Eile Edit View Project Tools Window H</td> <td>elp</td>	Eile Edit View Project Tools Window H	elp
Project Explorer # × HelloWorld_sw.c HelloWorld_hw.c Source Files 1 Sw @ HelloWorld_sw.c 1 HelloWorld_sw.c 3 HelloWorld_hw.c 4 HelloWorld_hw.c 4 <	: 🛺 😅 🕼 🖹 🚇 🎦 🖬 👗 🛍 🛍	「つ ペール が 榊 禁 神 🚽 茜 図 🕨 茜 🔍 🕲 🖉 🔷 💂
<pre> Application HelloWorld Application HelloWorld // Copyright (c) 2004, Impulse Accelerated Technologies, Inc. // Copyright (c) 2004, Impulse Accelerated Technologies, Inc. // All Rights Reserved. // All Rights Reserved. // Header Files // Header Files // Froject Files // Beadme.htm // Software process, defined in HelloWorld_sw.c // Software process, defined in HelloWorld_sw.c // Software process, defined in HelloWorld_sw.c extern void hear_hello(co_stream hello_in); li void say_hello(co_stream hello_out) is = { int i = 0; If _ SIM(cosim_logwindow log = cosim_logwindow_create("say_</pre>	Project Explorer 🛛 📮 🗙 🕞 H	elloWorld_sw.c 🕞 HelloWorld_hw.c
17 IF_SIM(cosim_logwindow log = cosim_logwindow_create("say_ 18 19 co stream open(hello out, 0 WRONLY, INT TYPE(32));		<pre>elloWorld_sw.c HelloWorld_hw.c 1 ///////////////////////////////////</pre>
		<pre>17 IF_SIM(cosim_logwindow log = cosim_logwindow_create("say_ 18 19 co stream open(hello out, 0 WRONLY, INT TYPE(32));</pre>

Files included in the HelloWorld project include:

Source files HelloWorld_sw.c and HelloWorld_hw.c - These source files represent the complete application, including the **main()** function, a software process, and a single hardware process.

Quartus subdirectory - Files in the Quartus subdirectory are used later in this tutorial to simplify the creation of the hardware platform.

Note

Due to limitations in the Altera SOPC Builder software, CoDeveloper projects must not be located in a directory that includes spaces in its name (such as "My Documents").

See Also

Step 2: Compiling the Application for Simulation

Tutorial 1: Hello World on the Nios II platform

1.2 Compiling the Application for Simulation

Hello World Tutorial for Nios II, Step 2

Before compiling the HelloWorld application to the target Nios II platform, let's first take a moment to understand its basic operation and the contents of its primary source files.

The specific process that we will be compiling to hardware is represented in HelloWorld_hw.c by the following function:

void say_hello(co_stream hello_out)

This hardware process simply sends data to an output stream (hello_out).

This hardware process is accompanied in the HelloWorld_hw.c file by a configuration function (**config_hello()**) that specifies the connection between the **hello_out** output stream and the corresponding input stream for the software process **hear_hello**.

The process **hear_hello** (which is declared as an **extern** function at the start of HelloWorld_hw.c) is defined in the source file HelloWorld_sw.c. This process will, when compiled, be loaded onto the Nios II processor and will communicate with the FPGA hardware via an automatically-generated hardware/software interface. In this case the software process simply reads data from the stream associated with the hardware process. (In a more typical Impulse C application there are multiple input/output streams associated with a given hardware process.)

HelloWorld_sw.c also includes a **main()** function for the application. This **main()** function makes reference to the lower-level configuration function **config_hello()** through the Impulse C standard function **co_initialize()**, which has been declared as an **extern** function and may be found in HelloWorld_hw.c.

Note: the organization of source files shown in this example is not required, but is recommended in order to simplify the compilation process. It is a good idea to keep the **main()** function and software processes of your application in one file or set of files (as shown), and to place hardware processes, configuration function, and **co_initialize()** in one or more different files. Doing so will avoid potential problems with cross-compiler incompatibilities between hardware and software processes as well as simplify the compilation and linking of complete applications for desktop simulation.

Simulating the Hello World Application

To compile and simulate the application for the purpose of functional verification:

1. Select Project -> Build Simulation Executable (or click the Build Simulation Executable button) to build the HelloWorld.exe executable. A command window will open, displaying the compile and link messages as shown below:

3

Build
====== Building target 'build_exe' in file_Makefile =======
"C:/Impulse/CoDeveloper3_307MinGW/bin/gcc" -g "-IC:\Impulse\CoDeveloper3_30\Include" ''-IC:/Impulse/CoDeveloper3_30/StageMaster/include" -DWIN32
"-IC:/Impulse/CoDeveloper3_30/MinGW/include" -o HelloWorld_hw.o -c HelloWorld_hw.c
"C:/Impulse/CoDeveloper3_30/MinGW/bin/gcc" -g "-IC:\Impulse\CoDeveloper3_30\Include" ''-IC:/Impulse/CoDeveloper3_30/StageMaster/include" -DWIN32
"-IC:/Impulse/CoDeveloper3_30/MinGW/include" -o HelloWorld_sw.o -c HelloWorld_sw.c
"C:/Impulse/CoDeveloper3_30/MinGW/bin/gcc" -g HelloWorld_hw.o HelloWorld_sw.o "C:\Impulse\CoDeveloper3_30\Libraries/ImpulseC.lib" -o HelloWorld.exe

====== Build of target 'build_exe' complete =======

4

 You now have a Windows executable representing the HelloWorld application implemented as a desktop (console) software application. Run this executable by selecting Project -> Launch Simulation Executable. A command window will open and the simulation executable will run as shown below:



Verify that the simulation produces the following output:

```
CPU listening for hello...
FPGA hardware says: 0 1 2 3 4 5 6 7 8 9
```

Note that, although the displayed messages indicate that the FPGA hardware has generated the indicated values, in fact the values are being generated by a hardware process which, for simulation purposes, has been compiled as a software process on your host development system (Windows). The next steps will show how to take this same Impulse C application and compile to actual hardware.

See Also

Step 3: Building the Application for the Target Platform

Tutorial 1: Hello World on the Nios II platform

1.3 Building the Application for the Target Platform

Hello World Tutorial for Nios II, Step 3

Specifying the Platform Support Package

The next step, prior to compiling and generating the HDL and related output files, is to select a platform

target. Which target you select has a number of implications, including:

- The output file format (e.g., VHDL, Verilog, or other intermediate language)
- The primitive components that will be referenced in the output (e.g., memory cores or buffer types)
- The types of optimizations performed during compilation
- The software library components that will be generated

To specify a platform target, open the Generate Options dialog as shown below (Project -> Options, Generate tab):

Options	\mathbf{X}
Build Simulate Generate System Registration	1
Altera Nios II (VHDL)	in the second
CoBuilder Optimization Options Enable constant propagation Scalarize array variables	
F Relocate loop invariant expressions	
Additional optimizer options:	Hardware build directory:
	hw
CoBuilder Generation Options Generate dual clocks Active-low reset Use std_logic types for VHDL interfaces	Software build directory: sw Hardware export directory: Quartus Software export directory:
Do not include co_ports in bus interface	Quartus
Library options:	
 Include floating point library Use higher latency, faster clock operators Allow double-precision types and operators 	
0	Cancel Apply Help

Specify **Altera Nios II (VHDL)** as shown. Also specify "hw" and "sw" for the hardware and software directories as shown, and specify "Quartus" for the hardware and software export directories. Click OK to save the options and exit the dialog.

Generate HDL for the Hardware Process

To generate hardware in the form of HDL files, and to generate the associated software interfaces and library files, select Generate HDL from the Project menu, or click on the Generate HDL button. A series of processing steps will run (in a command window) as shown below:

Build	$d \times$
Impulse C Software Interface Generator	
Copyright 2002-2007, Impulse Accelerated Technologies, Inc.	
All rights reserved.	
Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml	
Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml	
Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/system.xml	
Loading HelloWorld.xic	
for i in HelloWorld_sw.c; do cp \$i sw; done	
for i in ; do cp \$i sw; done	
chmod -R +rw sw	
======= Build of target 'build' complete =======	E

When processing is complete, several files will have been created in the "hw" and "sw" subdirectories of your project directory. Take a moment to review these generated files. They include:

Hardware directory ("hw")

6

- Generated VHDL source files (HelloWorld_comp.vhd, HelloWorld_top.vhd and subsystem.vhd) representing the hardware process and the generated hardware stream interface.
- A lib subdirectory containing required VHDL library elements.
- A class subdirectory containing generated files required by the Altera SOPC Builder tools.

Software directory ("sw")

- C source files extracted from the project that are required for compilation to the embedded processor (in this case HelloWorld_sw.c).
- A generated C file (co_init.c) containing the hardware configuration function. This file will also be compiled to the embedded processor.
- A **class** subdirectory containing additional software libraries to be compiled as part of the embedded software application. These libraries implement the software side of the hardware/software interface.

If you are an experienced Altera tools user, you may copy these files manually to your Altera project area and, if needed, modify them to suit your needs. In the next step, however, we will show how to use the hardware and software export features of CoDeveloper to move these files into your Altera project automatically.

See Also

Step 4: Exporting Files from CoDeveloper

Tutorial 1: Hello World on the Nios II platform

1.4 Exporting Files from CoDeveloper

Hello World Tutorial for Nios II, Step 4

As you saw in the previous step, CoDeveloper creates a number of hardware and software-related output files that must all be used to create a complete hardware/software application on the target platform (in this case an Altera FPGA with an embedded Nios II processor). You can, if you wish, copy these files manually and integrate them into your existing Altera projects. Alternatively, you can use the export features of CoDeveloper to integrate the files into the Altera tools semi-automatically. This section will walk you through the process, using a new Quartus project as an example.

Note: you must have the Altera Quartus II (version 4.0 or later) and SOPC Builder software installed in order to proceed with this and subsequent steps.

Options	
Build Simulate Generate System Registration	[
Platform Support Package: Altera Nios II (VHDL) CoBuilder Optimization Options Enable constant propagation Scalarize array variables Relocate loop invariant expressions Additional optimizer options:	Directories Hardware build directory:
CoBuilder Generation Options Generate dual clocks Active-low reset Use std_logic types for VHDL interfaces Do not include co_ports in bus interface Library options:	hw Software build directory: sw Hardware export directory: Quartus Software export directory: Quartus
Include floating point library Use higher latency, faster clock operators Allow double-precision types and operators	Cancel Apply Help

Recall that in <u>Step 3</u> you specified the directory "Quartus" as the export target for hardware and software:

These export directories specify where the generated hardware and software files are to be copied when the Export Software and Export Hardware features of CoDeveloper are invoked. Within these target directories (in this case we have specified both directories as "Quartus"), the specific destination (which may be a subdirectory under "Quartus") for each file is determined from the Platform Support Package architecture library files. It is therefore important that the correct Platform Support Package (in this case, Altera Nios II) is selected prior to starting the export process.

To export the files from the build directories (in this case "hw" and "sw") to the export directories (in this case the "Quartus" directory), select Project -> Export Generated Hardware (HDL) and Project -> Export Generated Software, or select the Export Generated Hardware and Export Generated Software buttons from the toolbar.

Build	
Impulse C HDL Design Generator	
Copyright 2002-2007, Impulse Accelerated Technologies, Inc.	_
Aurignis reserved. Li padring C/moules/CoDeveloper3_30/Architectures/altera_pios2.vml	
Loading C:/Impulse/CoDeveloper3_30/Architectures/HDL/Altera/Avalon/bus.xml	
Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/target.xml	
Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Altera/technology.xml	
Loading U:/Impulse/LoDeveloper3_3U/Architectures/VHDL/Generic/Generic/system.xml	
Colong reliave on a late "Colong lise/CoDeveloper3 30/bin/impulse_genvhdl.exe" HelloWorld.xhw.hw/HelloWorld_comp_vhd	
Impulse C RTL Component Generator	
Copyright 2002-2007, Impulse Accelerated Technologies, Inc.	
All rights reserved.	-
Generating say interview	
chmod R +rw hw	
"C:/Impulse/CoDeveloper3_30/bin/impulse_lib" ''-aC:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml" -hwdirhw -files "HelloWorld_sw.c"	
HelloWorld xic sw/co_init.c	
Impuise C souvere interace deneration Convinte 2002-2007, Impuise Accelerated Technologies, Inc.	
All rights reserved.	
Build	4 ×
Build Impulse C. Software Interface Generator	# ×
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc.	# ×
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved.	# ×
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml	* ×
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml	# ×
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/system.xml Loading HelloWorld.xic	× #
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/system.xml Loading HelloWorld.xic for i in HelloWorld_sw.c; do cp \$i sw; done	# ×
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/system.xml Loading C:Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/system.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/system.xml Loading C:Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/system.xml For i in HelloWorld_sw.c; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in ; do cp \$i sw; done For i in	× # •
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/system.xml Loading HelloWorld.xic for i in HelloWorld.sw.c; do cp \$i sw; done chmod-R + rw sw ''C:/Impulse/CoDeveloper3_30/Architectures.export'' -software -stodirsw ''-aC:/Impulse/CoDeveloper3_30/Architectures/altera_nios2 xml'' HelloWorld xic	* *
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/System.xml Loading HelloWorld.xic for i in HelloWorld.sw.c; do cp \$i sw; done for i in ; do cp \$i sw; done chmod -R +tw sw "C:/Impulse/CoDeveloper3_30/bin/impulse_export" -software -srcdirsw "-aC:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml" HelloWorld.xic "Quartus"	* ×
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/System.xml Loading HelloWorld.xic for i in HelloWorld.sw.c; do cp \$i sw; done for i in ; do cp \$i sw; done chmod -R +rw sw "C:/Impulse/CoDeveloper3_30/bin/impulse_export" -software -srcdirsw "-aC:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml" HelloWorld.xic "Quartus" Impulse C Design Exporter	# ×
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/System.xml Loading HelloWorld.xic for in HelloWorld_sw.c; do cp \$i sw; done for in ; do cp \$i sw; done chrimd. Fi +tw sw "C:/Impulse/CoDeveloper3_30/bin/impulse_export" -software -srcdirsw "-aC:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml" HelloWorld.xic "Quartus" Impulse C Design Exporter Copyright 2002-2007, Impulse Accelerated Technologies, Inc.	a × 2
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoD eveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoD eveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoD eveloper3_30/Architectures/VHDL/Generic/Generic/system.xml Loading G:/Impulse/CoD eveloper3_30/Architectures/VHDL/Generic/Generic/system.xml Loading HelloWorld.xic for in HelloWorld_sw.c; do cp \$i sw; done for in ; do cp \$i sw; done chrond-R1+rw sw "C:/Impulse/CoD eveloper3_30/bin/impulse_export" -software -srcdirsw "-aC:/Impulse/CoD eveloper3_30/Architectures/altera_nios2.xml" HelloWorld.xic "Quartus" Impulse C Design Exporter Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoD eveloper3_30/Architectures/altera_nios2.xml" HelloWorld.xic	a × 2
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoD eveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoD eveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoD eveloper3_30/Architectures/VHDL/Generic/Generic/System.xml Loading G:/Impulse/CoD eveloper3_30/Architectures/VHDL/Generic/Generic/System.xml Loading G:/Impulse/CoD eveloper3_30/Architectures/VHDL/Generic/Generic/System.xml Loading G:/Impulse/CoD eveloper3_30/Architectures/VHDL/Generic/Generic/Sestem.xml Loading G:/Impulse/CoD eveloper3_30/Architectures/VHDL/Generic/Generic/Sestem.xml Loading HelloWorld_sw.c; do cp \$i sw; done for in r do cp \$i sw; done chroid -R +rw sw "C:/Impulse/CoD eveloper3_30/bin/impulse_export" -software -srcdirsw "-aC:/Impulse/CoD eveloper3_30/Architectures/altera_nios2.xml" HelloWorld.xic "Quartus" Impulse C Design Exporter Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoD eveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoD eveloper3_30/Architectures/Altera/NiosII/cpu.xml	
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/System.xml Loading HelloWorld.xic for i in HelloWorld_sw.c; do cp \$i sw; done fori in; i do cp \$i sw; done chmod-R +rw sw "C:/Impulse/CoDeveloper3_30/bin/impulse_export" -software -srcdirsw "-aC:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml" HelloWorld.xic "Quartus" Impulse C Design Exporter Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml	# × ₹
Build Impulse C Software Interface Generator Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera/NiosII/cpu.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/VHDL/Generic/Generic/system.xml Loading HelloWorld.xic for i in HelloWorld_sw.c; do cp \$i sw; done fori in ; do cp \$i sw; done chmod.F1 +tw sw "C:/Impulse/CoDeveloper3_30/bin/impulse_export" -software -srcdirsw "-aC:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml" HelloWorld.xic "Quartus" Impulse C Design Exporter Copyright 2002-2007, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3_30/Architectures/altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera_nios2.xml Loading C:/Impulse/CoDeveloper3_30/Architectures/Altera_nios2.xml Loadi	# × ₹

Note: you must select BOTH Export Software and Export Hardware before going onto the next step.

You have now exported all necessary files from CoDeveloper to the Quartus project directory.

See Also

8

Step 5: Creating a Quartus Project

Tutorial 1: Hello World on the Nios II platform

1.5 Creating a Quartus Project

Hello World Tutorial for Nios II, Step 5

Now we'll move into the Altera tool environment. Begin by launching Altera Quartus II (from the **Windows Start -> All Programs -> altera -> Quartus II 8.0 -> Quartus II 8.0 (32-Bit)**). Open a new project by selecting **File -> New Project Wizard**.



In the field prompting you for the new project's working directory, use the browse button and find the directory (**Quartus**) to which you exported the hardware and software files in the previous step:

9



Select the **Quartus** directory and click **Open**. On page one of the **New Project Wizard** dialog, enter **HelloWorld** in both the project name and top-level design entity fields as shown:

D. Vallera vinipuiset xa	mples\HelloWo	rld_NIOS\Quart	us		
What is the name of t	nis project?				
HelloWorld	iis project:				
What is the name of the entite th	ne top-level desi ty name in the d	gn entity for this esign file.	s project? This n	ame is case sensi	itive and must
HelloWorld					
Use Existing Project	Settings				

Now click Next.

Now you will import the VHDL files generated by CoDeveloper to your Quartus project. In the Add Files page (page 2), add the files in the following order:

- 1. Core logic files in the user_logic_HelloWorld_arch_module subdirectory: subsystem.vhd, HelloWorld_top.vhd and HelloWorld_comp.vhd
- 2. All .vhd files in the impulse_lib subdirectory

The files should be listed in the opposite order from which they were added (e.g. the **impulse_lib** files should be at the top of the list):

File name		Туре	Add All
impulse_lib/av impulse_lib/cr/	alon_it.vhd agister.vhd	VHDL File VHDL File	Remove
impulse_lib/cr	ianal vhd	VHDL File	11011070
impulse lib/div	vmod. vhd	VHDL File	Properties
impulse_lib/fife	p.vhd	VHDL File	
impulse_lib/fife	o_dc.vhd	VHDL File	Up
impulse_lib/gr	nem.vhd	VHDL File	
impulse_lib/imp	pack.vhd	VHDL File	Down
impulse_lib/me	em_if.vhd	VHUL File	
impulse_lib/se	eam yhd		
impulse_lib/str	eam do vhd	VHDL File	
user logic He	lloWorld arch module/HelloWorld comp.vhd	VHDL File	
user_logic_He	lloWorld_arch_module/HelloWorld_top.vhd	VHDL File	
user_logic_He	lloWorld_arch_module/subsystem.vhd	VHDL File	
<		>	
necifu the nath	names of any non-default libraries	as Liberation	

Click Next to proceed.

In the **Family and Device Settings** page (page 3) select the device you will be targeting. For this example we will choose **Cyclone III**, **EP3C25F324C6** device, with 324 pins and speed grade 6. This is the FPGA used in the **Cyclone III FPGA Evaluation Kit**.

				- Show in 'Ava	ailable dev	/ice' list —
Family: Cyclone I			-	Package:	Any	•
Devices: All			-	Pin count:	324	
-			_	Speed grade	. 6	-
Target device				speed grade: o		
Auto device sele	cted by the Fitter			I✓ Show ad	lvanced d	levices
 Specific device s 	elected in 'Availa	ble devices	' list	HardCop	y compati	ible only
unione:						
	Contra	1.5.	1112	Manage	Table J	DU
Name Eponoseooxine	1 27	24604	User 17	COODEC	100 LTDE	I FLL
EP3C40F324C6	1.2V	39600	196	1161216	252	4
-						
		100				
×						
Companion device —						
Companion device — HardCopy:						

Click Next again. Skip the EDA Tool Settings page (page 4) by again clicking Next.

You will see a **Summary** page listing the options you have chosen as shown below:

D:/altera/ImpulseExamples/I	HelloWorld_NIOS/Quartus/
Project name:	HelloWorld
Top-level design entity:	HelloWorld
Number of files added:	15
Number of user libraries added:	0
Device assignments:	
Family name:	Cyclone III
Device:	EP3C25F324C6
EDA tools:	
Design entry/synthesis:	<none></none>
Simulation:	<none></none>
Timing analysis:	<none></none>
Operating conditions:	
VCCINT voltage:	1.2V
Junction temperature range:	0-85 °C

Click Finish to exit the Wizard and return to Quartus.

The next steps will demonstrate how to create and configure a hardware system with a Nios II processor and the necessary I/O interfaces for our sample application.

See Also

Step 6: Creating the New Platform

Tutorial 1: Hello World on the Nios II platform

1.6 Creating the New Platform

Hello World Tutorial for Nios II, Step 6

Now that you have created a **Quartus** project using the wizard, you will need to specify additional information about your platform in order to support the requirements of your software/hardware application. These steps include the creation of a hardware system with a Nios II processor and the necessary I/O elements.

You will use **SOPC Builder** to create a hardware system containing an Altera Nios II embedded processor, the FPGA module created for the HelloWorld hardware process by CoBuilder, and several necessary peripherals. To do this, select **Tools** -> **SOPC Builder** to start SOPC Builder.

A new **SOPC Builder** window will appear. In the **Create New System** dialog that appears, enter **HelloWorldSystem** as the **System Name**, and specify **VHDL** for the **Target HDL** Language:

Greate New System - Altera SC File Edit Module System View T)PC Builder – unnamed.sopc (D:\a ools Help				d.sopc) 📘 🗖 🗙
System Contents System Generation					
Altera SOPC Builder Create new component Nios II Processor Bridges and Adapters Interface Protocols Legacy Components	Target Device Family: Cyclone III	Clock Settings Name	Source	MHz	Add
H-Legacy Components Memories and Memory Controllers Peripherals PLL Use Md Create New System System Name; HelloWorldSystem System Name; HelloWorldSystem Target HDL: OVerlog OK Cancel					End
Add	Remove Edit	Move Up	Move Down	Address Map	Filter
Info: Your system is ready to generate	e. Exit Help	Prev Next	Generate		

Click OK to continue.

Note: the System Name that you specify in this step must be a valid VHDL identifier: specifically, it must not begin with a numeric character or include spaces or other non-alphanumeric characters other than the underscore character (_).

See Also

Step 7: Configuring the New Platform

Tutorial 1: Hello World on the Nios II platform

1.7 Configuring the New Platform

Hello World Tutorial for Nios II, Step 7

The following instructions will lead you through the process of creating your Nios II-based platform using SOPC Builder. This process requires many steps, but will only need to be done once for each new project that you create.

We'll begin by adding the largest component of the **HelloWorld** system, the Nios II processor. From the **System Contents** tab (on the left side of the **SOPC Builder** window), double-click **Nios II Processor** under **Altera SOPC Builder**. The **Nios II Processor - cpu** configuration Wizard will appear. Select the **Nios II/s** core as shown below:

🖳 Nios II Processor	- сри				\mathbf{X}
Mios MegaCore	II Processor				Documentation
Parameter Settings Core Nios II Cach	es and Memory Interfaces	Advanced Features	MMU and MPU Settings > JTA	G Debug Module >	Custom Instructions
Select a Nios II core:					
	ONios II/e	Nios II/s	○Nios II/f		
Nios II Selector Guide Family: Cyclone III f _{system:} 50.0 MHz cpuid: 0	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		
Performance at 50.0 MH	z Up to 8 DMIPS	Up to 32 DMIPS	Up to 57 DMIPS		
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs		
Memory Usage	Two M9Ks (or equiv.)	Two M9Ks + cache	Three M9Ks + cache		
Hardware Multiply: Emb	edded Multipliers	Hardware Divide			

Click **Finish** to add the Nios II CPU to the system and return to **SOPC Builde**r. A module called **cpu** appears in the SOPC window.

Altera SOPC Builder - HelloWo File Edit Module System View T	ortdSyst 'ools Ni	ternisoj ios II He	c* (D:\altera\lmpul p	seExamples\Hello	World_NIOS\Quart	us\HelloWor	ldSystem.sopc)		
System Contents System Generation									
🙀 Attera SOPC Builder	Targ	et	3	Clock Settings					
Create new component	Device Family: Cyclone III		Name Source clk External			MHz		Add	
Bridges and Adapters Interface Protocols Legacy Components Memories and Memory Controllers					al 50.0			Remove	
⊞-Peripherals	Use Con Module Name		Module Name	Description		Clock	Base	End	IRG
			🗆 cpu	Nios II Process	or	1			
e User Logic			instruction_maste data_master jtag_debug_modu	r Avalon Memor Avalon Memor Ile Avalon Memor	y Mapped Master y Mapped Master y Mapped Slave	clk	IRQ 0x0000080	0 0 0x00000	IRQ 31↔ fff

Next, you must add the necessary peripherals to the new Nios II system. If you are not familiar with how to do this in **SOPC Builder**, you may wish to review the information provided in your **Nios II Development Kit** documents, and in particular the tutorials provided by Altera. Refer to the instructions provided by Altera in the following file:

http://www.altera.com/literature/tt/tt_nios2_hardware_tutorial.pdf

Using the methods described in the Altera documentation (and summarized below), you will need to add the following components:

On-Chip Memory

To add the on-chip memory peripheral, **onchip_mem**, perform the following steps:

Locate Memories and Memory Controllers -> On-Chip -> On-Chip Memory (RAM or ROM) and double-click to add. The On-Chip Memory (RAM or ROM) - onchip_mem wizard displays.

In the Size box, type in 20 as memory size, and make sure the unit is KBytes:

😫 Edit Module - Altera SOPC Builde	💶 On-Chip Memory (RAM or ROM) - onchip_mem	
File Edit Module System View Tools System Contents System Generation Attern SOPC Builder	On-Chip Memory (RAM or ROM)	Documentation
Create new component Nios II Processor Bridges and Adapters	Parameter Settings	
Interface Protocols Identified Protocols <p< td=""><td>Memory type RAM (Writable) Dual-port access Read During Write Mode: DONT_CARE Block type: Auto Initialize memory content Memory will be initialized from onchip_mem.hex Size Data width: 32 Total memory size: 32 KBytes</td><td></td></p<>	Memory type RAM (Writable) Dual-port access Read During Write Mode: DONT_CARE Block type: Auto Initialize memory content Memory will be initialized from onchip_mem.hex Size Data width: 32 Total memory size: 32 KBytes	
Add	Read latency Slave s1: 1 Memory initialization Enable non-default initialization file User-created initialization file: Onchip_mem Instance ID: NONE	
		Cancel Finish

Click Finish. You are returned to the Altera SOPC Builder window.

JTAG UART Interface

The **JTAG UART** is used for communication between the board and the host machine and for debugging software running on the Nios II processor. To add the **JTAG UART** peripheral, **jtag_uart**, perform the following steps:

Locate Interface Protocols -> Serial -> JTAG UART, and double-click to add. The JTAG UART - jtag_uart wizard displays.

File Edit Module System View Tool System Contents System Generation Document System Contents System Generation Document Altera SOPC Builder Parameter Settings Nios II Processor Simulation Simulation High Speed FOI Setial PCI Construct using registers instead of memory blocks 8	
System Contents System Generation Document Image: Contract Source So	
Attera SOPC Builder Parameter Create new component Settings Nios II Processor Configuration Interface Protocols Configuration ASI Buffer depth (bytes): High Speed IRQ threshold: PCI Construct using registers instead of memory blocks	ation
Avalor.ST KAG Interface Avalor.ST KAG Interface Avalor.ST Serial Peripheral SPI (3 Wire Serial) UART (RS-232 Serial Port)	Finish

Leave all options at their default settings.

Click Finish. You are returned to the Altera SOPC Builder window.

Adding the Hardware Process Module "HelloWorld_arch"

Now add the **HelloWorld_arch** module, which implements the HelloWorld hardware process. Locate **User Logic** -> **HelloWorld_arch module** and double-click to add:



You have now completed adding the necessary peripherals. The modules and their associated names should appear as shown below:

Device Family: Cyclone III 🛛 Nar		Cyclone III 🛛 🔽	Name	Source		MHz		bh4
			cik	External		50.0	Re	move
Use	Con	Module Name	1	Description	Clock	Base	End	IRQ
		🗆 сри		Nios II Processor				
		instruction_master data_master jtag_debug_module	3	Avalon Memory Mapped Master Avalon Memory Mapped Master Avalon Memory Mapped Slave	clk	IRQ	0 IRQ 3 0x00000fff	ı
>		onchip_mem s1		On-Chip Memory (RAM or ROM) Avalon Memory Mapped Slave	cik	■ 0x00002000	0x00006fff	
		☐ jtag_uart avalon_itag_slave		JTAG UART Avalon Memory Mapped Slave	cik	■ 0x0000000	0x0000007	
¥		user_logic_HelloV	/orld_arch_module_classic_0	HelloWorld_arch module	clk	- 000000010	0×0000001 f	

Editing CPU Settings

Now double-click the **cpu** module to edit its settings. Change the Memory for both Reset Vector and Exception Vector to **onchip_mem**, as shown:

Mios MegaCore	s II Processor				Documentation
Parameter Settings					
Core Nios II 🔪 Cac	hes and Memory Interfaces $>$	Advanced Features	MMU and MPU Settings	> JTAG Debug Module	Custom Instructions
Core Nios II					
Select a Nios II core:					
	ONios II/e	Nios II/s	○Nios II/f	1	
Nios II Selector Guide Family: Cyclone III f _{system:} 50.0 MHz cpuid: 0	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 Pred	liction	
Performance at 50.0 MH	Hz Up to 8 DMIPS	Up to 32 DMIPS	Up to 57 DMIPS		
Logic Usage Memory Usage	600-700 LEs Two M9Ks (or equiv.)	1200-1400 LEs Two M9Ks + cache	1400-1800 LEs Three M9Ks + cache		
Hardware Multiply: Emi	oedded Multipliers	Hardware Divide			
Reset Vector: Me	mory: onchip_mem	Offset: 0x0	c	×00002000	
Exception Vector: Mer	nory: lonchin mem	Offset: 0v20	0	/00002020	

Click OK to save the changes.

Assigning Addresses

20

Next, select the SOPC Builder menu **System** -> **Auto-Assign Base Addresses**. This will automatically assign addresses to each memory-based module. Address space conflict can be avoided. The system will look like the following after the address assigning is done.

Use	Con	Module Name	Description	Clock	Base	End	IRQ
		🗆 cpu	Nios II Processor				
	F	instruction_master data_master jtag_debug_module	Avalon Memory Mapped Master Avalon Memory Mapped Master Avalon Memory Mapped Slave	clk	IRQ 0 0x00010800	IRQ 31 0x00010fff	
	$ \downarrow \downarrow$	ightarrow on chip_mem s1	On-Chip Memory (RAM or ROM) Avalon Memory Mapped Slave	cik	≓ 0x00008000	0x0000cfff	
		jtag_uart avaion_itag_slave	JTAG UART Avalon Memory Mapped Slave	cik	₩ 0x00011010	0x00011017	
		user_logic_HelloWorld_arch_module_classic_0 p_hear_hello_hello_in	HelloWorld_arch module Avalon Memory Mapped Slave	cik	■ 0x00011000	0x0001100f	

Save the system settings by selecting from the menu File -> Save.

Your new Nios II platform is ready for system generation.

See Also

Step 8: Generating the System

Tutorial 1: Hello World on the Nios II platform

1.8 Generating the System

Hello World Tutorial for Nios II, Step 8

At this point you have set up and configured your new Nios II-based platform, including the hardware module generated by CoBuilder, and can now start the system generation process within SOPC Builder.

Click **Generate** on the bottom of the **SOPC Builder** window to generate the system. The **SOPC Builder** will automatically switch to the **System Generation** tab and display generation information.Make sure the **Simulation** option is unchecked to save time. This process may take several minutes.

ڬ Altera SOPC Builder - HelloWorldSystem.sopc (D:\altera\ImpulseExamples\HelloWorld_NIOS\Quartus\Hel 🔳	
File Edit Module System View Tools Nios II Help	
System Contents System Generation	
Options System module logic will be created in VHDL. Simulation. Create project simulator files. Run Simulator	
Nios II Tools Nios II IDE	
0 Info: Quartus II Shell was successful. 0 errors, 0 warnings	
Info: Peak virtual memory: 47 megabytes	
Info: Processing ended: Wed Feb 18 15:46:19 2009	
Info: Elapsed time: 00:00:00	
Info: Total CPU time (on all processors): 00:00:00	
# 2009.02.18 15:46:19 (*) Completed generation for system: Hello/VorldSystem.	
# 2009.02.18 15:46:19 (*) THE FOLLOWING SYSTEM ITEMS HAVE BEEN GENERATED:	
SOPC Builder database : D:/altera/ImpulseExamples/HelloWorld_NIOS/Quartus/HelloWorldSystem.ptf	
System HDL Model : D:/altera/impulseExamples/HelloWorld_NIOS/Quartus/HelloWorldSystem.vhd	
System Generation Script : D:/altera/ImpulseExamples/Hello/Vorld_NIOS/Quartus/Hello/VorldSystem_generation_script	
# 2009.02.18 15:46:19 (*) SUCCESS: SYSTEM GENERATION COMPLETED.	
Info: System generation was successful.	
	>
Info: Your system is ready to generate.	
Exit Help Prev Next Cenerate Generate	

When generation is complete you may exit SOPC Builder and return to Quartus.

Creating a Block Diagram

Now you will need to create a block diagram to connect the complete SOPC Builder-generated system (which includes the HelloWorld hardware process module, the Nios II processor, and peripherals) to the pins on the FPGA. Open a **File** -> **New** dialogue from the Quartus menu, and select **Block Diagram/Schematic File** as shown below:



Click OK, and a new DBF file will appear. Now add the block representing the SOPC Builder-generated system. Double-click anywhere in the new block diagram file to bring up the **Symbol** dialog. Open the **Project** folder and select the **HelloWorldSystem** symbol as shown below:



Click OK, and a symbol outline appears attached to the mouse pointer. Drag the symbol to a proper location in the Block Diagram.



Next, bring up the **Symbol** dialog again, and select **input** port as shown below:



Click OK to add the input port to the block diagram.

Next, bring up the Symbol dialog again, and select vcc as shown below:



Click OK to add the vcc to the block diagram. Align the **input** port with the **clk** pin, and the vcc to the **reset_n** pin on the block diagram as shown:



Change the name of the **input** port by double-clicking it, and then type in the name **osc_clk** in the **Pin name** box.

© 2003-2009 Impulse Accelerated Technologies



Click OK to accept the change.

Pin Assignment

The next step is to assign pins. Here we only have one pin that needs assignment. Right-click the **osc_clk** pin, and select **Locate in Assignment Editor**.

			HelloWorldSystem	
psc_clk		> INERT	clk	
			K Cut Ctrl+	x ::::::::::::::::::::::::::::::::::::
			Ctrl+	c
			🔁 Paste 🛛 🛛 Ctrl+	v
· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · ·		🗙 <u>D</u> elete De	el
· · · · · · · · · · · · · · · · · · ·		· · · · · · · · ·	Locate	Locate in Assignment Editor
			Update Symbol or Block	Locate in <u>P</u> in Planner
			△▲ Flip Horizontal	K Locate in Chip Planner (Floorplan & Chip Editor)
		:::::	🗧 Flip <u>V</u> ertical	Locate in <u>R</u> esource Property Editor
			Rotate <u>b</u> y Degrees	Locate in Technology Map Viewer
			700m	Locate in RTL Viewer
::::::		:::::	20011	Locate in Design File
		:::::	Properties	

The Assignment Edit window will appear with the **osc_clk** pin:

· Category: 지	All Locations PLL Comb. cell Register cell Embedded multiplier Clock control block Impedance control b Oscillator block	block lock				
Node Filter:	Show assignments for	specific nodes:				
Informati	This cell specifies the des Node Finder to assign a d	ination name for poi estination name.	int-to-point assignme	ents. For single-point as	signments, this cell	specifies the destination of the
	Edit: XV	sc_clk				
7	From	То		Assignment Name	Value	Enabled
1		🔷 osc_clk				Yes

In the **Assignment Name** column, choose **Location** from the list. In the **Value** column, type in **V9** as the pin location.

7	From	То	Assignment Name	Value	Enabled
1		osc_clk	Location	V9	Yes
2		osc_clk	10.		Yes

Save your project. Save the block diagram file as **HelloWorld.bdf**. This will be your top-level design file for the **HelloWorld** project.

Your project is now ready for bitmap generation and subsequent downloading.

Tip: you may wish to to save your Altera project at this point and save a copy for later use with other CoBuilder-generated projects.

See Also

28

Step 9: Generating the FPGA Bitmap

Tutorial 1: Hello World on the Nios II platform

1.9 Generating the FPGA Bitmap

Hello World Tutorial for Nios II, Step 9

At this point, if you have followed the tutorial steps carefully you have successfully:

- Generated hardware and software files from the CoDeveloper environment.
- Created a new Altera Quartus II project and used SOPC Builder to create a new Nios II-based platform.
- Imported your CoDeveloper-generated files to the Altera tools environment.
- Completed a block diagram and assigned pins for the selected FPGA device.

You are now ready to compile your design, generate the bitmap and download the complete application to the target platform. This process is not complicated (at least in terms of your actions at the keyboard) but can be quite time consuming due to the large amount of processing that is required within the Altera tools.

To generate the bitmap, select **Processing -> Start Compilation** as shown below:



Note: this process may require a few minutes to complete, depending on the speed and memory of your development system.

During compilation, Quartus will analyze the generate VHDL source files, synthesize the necessary logic, and create logic that is subsequently placed and routed into the FPGA along with the Nios II processor and interface elements that were previously specified. The result will be a bitmap file (in the appropriate Altera format) ready for downloading to the device.

When the bitstream has been generated, select **Tools** -> **Programmer** to open the new programming file. Select **File** -> **Save As** and save the chain description file as **HelloWorld.cdf** (make sure the **Add file to current project** option is selected).

The programming file HelloWorld.sof should be visible in the programming window. If it is not, select **Add File...** and open **HelloWorld.sof**.

Enable Program/Configure for HelloWorld.sof and make sure your programming hardware (e.g., the **USB-Blaster [USB-0]** cable) is configured properly. Click **Start** to begin downloading the **HelloWorld.sof** file to the target device.

Note: If you don't have the full license for **OpenCore Plus** megafunctions, then a message will pop up. Click **OK** to continue. The bitmap file with be named **HelloWorld_time_limited.sof**. After the downloading is done, a **OpenCore Plus Status** message box will pop up. Don't click the **Cancel** button. Otherwise the downloaded bitmap will be reset.

💾 Quartus II -	D:/altera/impulseExam	ples/HelloWorld	_NIOS/Quartus/H	lelloWorld -	HelloWorld -	[Hello)	Norld_t	ime_L	. 🗆 🗙
File Edit Proces	sing Tools Window	<i></i>							
🔔 Hardware Set	up USB-Blaster [USB-0]		Mode	JTAG	•	Progre	ss:	0%	
Enable real-time	e ISP to allow background pro	gramming (for MAX II	devices)						
M Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit
1 Stop	HelloWorld_time_limited.	EP3C25F324	003490A2	FFFFFFFF					
Auto Detect									
X Delete	<u>i</u> l								
📛 Add File									
避 Change File									
Save File									
😂 Add Device									

Now that the hardware is programmed, you are ready to download and run the software application on the platform.

See Also

Step 10: Running the Application on the Platform

Tutorial 1: Hello World on the Nios II platform

1.10 Running the Application on the Platform

Hello World Tutorial for Nios II, Step 10

In the previous step, you programmed the FPGA device with the design you created in **Quartus** and **SOPC Builder**. Now you will use **Altera Nios II IDE** to compile the software portion of the project and run it on the development board.

Begin by starting the Nios II IDE (usually available in the Windows Start menu -> All Programs -> altera -> Nios II EDS 8.0 -> Nios II 8.0 IDE). If the Workspace Launcher dialog box appears, click OK to use the default workspace.

A **Nios II IDE** window will appear. To create a new project to manage the **HelloWorld** software files, select **File** -> **New** -> **Project...**; the following wizard will appear:

📑 New Project	
Select a wizard The Nios II IDE will create and manage the makefile to build this application project.	-
Wizards:	har and the second seco
type filter text Nos II C/C++ Application General Altera Nios II General Gener	
(?) < Back Next > Finish (Cancel

Select Altera Nios II C/C++ Application as shown, and click Next.

On the next page, select the project path, target hardware, and project template as follows, using the Browse buttons to locate the appropriate **Path** and **SOPC Builder System** options:

Name: HelloWorld

Specify Location: checked

Path: D:\altera\ImpulseExamples\HelloWorld_NIOS\Quartus\software\HelloWorld_arch (The project path should point to the software files that were exported by CoDeveloper in (

<u>Step 4</u>.)

SOPC Builder System:

D:\altera\ImpulseExamples\HelloWorld_NIOS\Quartus\HelloWorldSystem.ptf (This is the system .ptf file generated by SOPC Builder in <u>Step 8</u>.)

CPU: cpu

Select Project Template: Blank Project

Impulse HelloWorld Tutorial for Altera CyclonellI Evaluation Kit

32

Name: HelloWorld	
Specify Location	
Location: D:\altera\ImpulseE	xamples\HelloWorld_NIOS\Quartus\software Browse
Select Target Hardware.	
SOPC Builder System PTF File:	a\ImpulseExamples\HelloWorld_NIOS\Quartus\HelloWorldSystem.ptf 💟 Browse
CPU:	cpu
Select Project Template	
Blank Project	Description
Board Diagnostics Count Binary Hello Freestanding Hello MicroC/OS-II Hello World Hello World Small Memory Test Simple Socket Server Web Server	Creates a blank project
	Details
	Blank Project creates an empty project to which you can add your code. This software example runs on the following Nios II hardware designs:

Click **Finish** to create the new project. Two new projects (**HelloWorld** and **HelloWorld_syslib**) should appear in the **Nios II C/C++ Projects** window in the Nios II IDE. Copy the software files that were exported in <u>Step 4</u> (**co_init.c** and **HelloWorld_sw.c**) to the **HelloWorld** project as shown below.

Nios II C/C++ - Nios II IDE
File Edit Navigate Search Project Tools Run Window Help
📑 • 🚍 🔄 💼 🕴 🎯 • 🞯 • 🞯 • 🛛 🕸 • 🕥 • 💁 - 🛛 🥔 •
Nios II C/C++ Projects 🕱 🦳 🗖 🗖
(
General Components General Component
🖾 HelloWorld_axch
File Edit View Favorites Tools Help
G Back + 🕥 - 🎓 Search 🔊 Folders 💷 -
Address 🛅 D:\altera\ImpulseExamples\HelloWorld_NIOS\Quartus\software\HelloWorld_arch
Folders × Name ▲
□ □

The software files will appear under the HelloWorld project as shown below:



Before you compile the project, you need to adjust some of the project settings to minimize the size of the executable file in order to fit the 20 KBytes onchip memroy. Right-click **HelloWorld** and select **System Library Properties**. The **Poperties** dialog box of **HelloWorld_syslib** will appear.

Change to following settings:

- check Programs never exits
- uncheck Support C++
- uncheck Clean exit (flush buffers)

- check Reduced device drivers
- check Small C library

Info Builders C/C++ Build C/C++ Build C/C++ File Types C/C++ Include Paths and C/C++ Indexer C/C++ Make Project C/C++ Project References Refactoring History System Library

Click **OK** to save the changes.

Now build the project by right-clicking the **HelloWorld** project and selecting **Build Project**. The IDE will build the **HelloWorld_syslib** system library, which includes a driver for the Impulse C hardware module created by **CoBuilder**, along with the application software code in the **HelloWorld** project.

Build Project
Operation in progress
Building HelloWorld
Run in Background Cancel Details >>
Problems 📃 Console 🛛 Properties
C-Build [HelloWorld]
**** Build of configuration Debug for project HelloWorld ****
make -s all includes
Compiling HelloWorld sw.c
Compiling co_init.c
Linking HelloWorld.elf

Once the software has finished building, you are ready to run the application on the hardware platform. Right-click the **HelloWorld** project and select **Run As -> Nios II Hardware**. You should see the following output in the Console window:

Problems 📮	Console 🕴 Properties
HelloWorld Nios II	HW configuration [Nios II Hardware] Nios II Terminal Window (2/19/09 12:03 PM)
nios2-termi	nal: connected to hardware target using JTAG UART on cable
nios2-termi	nal: "USB-Blaster [USB-0]", device 1, instance O
nios2-termi	nal: (Use the IDE stop button or Ctrl-C to terminate)
CPU listeni	ng for hello
FPGA hardwa	are says: 0 1 2 3 4 5 6 7 8 9

Compare the output to that of the desktop simulation done in <u>Step 2</u>--they should be the same.

Congratulations! You have successfully compiled a complete hardware/software application on the target FPGA device, without the need to write any low-level HDL code.

Although this sample application was trivial (in terms of the amount of logic generated on the FPGA), the process for substantially larger, more complex applications is essentially the same. You can follow these steps when compiling your own Impulse C applications.

See Also

Tutorial 1: Hello World on the Nios II platform