

2. Tour of the Nios II IDE

NII52002-6.0.0

Introduction

This chapter familiarizes you with the main features of the Nios II integrated development environment (IDE). This chapter is only a brief introduction to the look-and-feel of the Nios II IDE—it is not a user guide. The easiest way to get started using the Nios II IDE is to launch the tool and perform the Nios II software development tutorial, available in the help system.

Because of evolution and improvement of the software, the figures in this chapter may not match exactly what you see in the actual software.



For more information on all IDE-related topics, refer to the Nios II IDE help system.

The Nios II IDE Workbench

The term "workbench" refers to the desktop development environment for the Nios II IDE. The workbench is where you edit, compile and debug your programs. Figure 2–1 shows an example of the workbench.

Figure 2–1. The Nios II IDE Workbench



Perspectives, Editors & Views

Each workbench window contains one or more perspectives. Each perspective provides a set of capabilities aimed at accomplishing a specific type of task. For example, Figure 2–1 shows the C/C++ development perspective.

Most perspectives in the workbench comprise an editor area and one or more views. An editor allows you to open and edit a project resource (i.e., a file, folder, or project). Views support editors, provide alternative presentations, and ways to navigate the information in your workbench. Figure 2–1 shows a C program open in the editor, and the C/C++ Projects view in the left-hand pane of the workbench. The C/C++ Projects view displays information about the contents of open projects.

Any number of editors can be open at once, but only one can be active at a time. The main menu bar and toolbar for the workbench window contain operations that are applicable to the active editor. Tabs in the editor area indicate the names of resources that are currently open for editing. An asterisk (*) indicates that an editor has unsaved changes. Views can also provide their own menus and toolbars, which, if present, appear along the top edge of the view. To open the menu for a view, click the drop-down arrow icon at the right of the view's toolbar or right-click in the view. A view may appear on its own, or stacked with other views in a tabbed notebook.

Creating a New Project

The Nios II IDE provides a **New Project** wizard that guides you through the steps to create a new C/C++ application project. To start the C/C++ application **New Project** wizard, click File, New, **C/C++ Application**. See Figure 2–2.

C/C++ - hello_world.c - Nios II IDE File Edit Navigate Search Run Project Tools Window Help New Alt+Shift+N 🔹 🕨 🃑 Project... Ctrl+F4 Close 😂 Folder Close All Ctrl+Shift+F4 📑 File Ctrl+S 🖳 Save ท Header File 😸 Save As... C/C++ Application Save All Ctrl+Shift+S 📑 Other... Chrl+N Revert

Figure 2–2. Starting the C/C++ Application New Project Wizard

The C/C++ application **New Project** wizard prompts you to specify:

- 1. A name for your new project.
- 2. The target hardware.
- 3. A template for the new project.

Project templates are ready-made, working designs that serve as examples to show you how to structure your own projects. It is often easier to start with a working "Hello World" project, than to start a blank project from scratch.

Figure 2–3 shows the C/C++ application **New Project** wizard, with the template for a Dhrystone benchmark design selected.

/erilog\niosII_stratix_1s10\standard\software' Browse,,,
piosTI strativ_1s10)standard)std_1s10.ptf - Browse
_
scription
uns the Dhrystone 2.1 benchmark code
hrystone uses the 2.1 version of the Dhrystone
when run on the fast hardware example design on a
ou are running this program on a hardware design
ther than the fast example design, you must change
our read-only, read-write, and program memory to

Figure 2–3. The C/C++ Application New Project Wizard

After you click **Finish**, the Nios II IDE creates the new project. The IDE also creates a system library project, ***_syslib** (for example, **dhrystone_0_syslib** for Figure 2–3). These projects show up in the **C/C++ Projects** view of the workbench.

Building & Managing Projects

Right-clicking on any resource (a file, folder, or project) opens a contextsensitive menu with operations you can perform on the resource. Rightclicking is usually the quickest way to find the operation you need, though operations are also available in menus and toolbars.

To compile a project, right-click the project in the **C/C++ Projects** view, and choose **Build Project**. Figure 2–4 shows the context-sensitive menu for the project dhrystone_0, with the **Build Project** option chosen. When building, the Nios II IDE first builds the system library project (and any other project dependencies), and then compiles the main project. Any warnings or errors are displayed in the **Tasks** view.



Figure 2–4. Building a Project Using the Context-Sensitive (Right-Click) Menu

Right-clicking a project in C/C++ Projects also allows you to access the following important options for managing the project:

- Properties—Manage the dependencies on target hardware and other projects
- System Library Properties—Manage hardware-specific settings, such as communication devices and memory partitioning
- **Build Project**—i.e., make
- **Rebuild Project**—i.e., make all
- **Run As**—Run the program on hardware or the ISS
- **Debug As**—Debug the program on hardware or the ISS

Running & Debugging Programs

Run and debug operations are available by right-clicking the project. The Nios II IDE allows you to run or debug the project either on a target board or the Nios II instruction set simulator (ISS). For example, to run the program on a target board, right-click the project in the **C/C++ Projects** view, then click Run As, **Nios II Hardware**. See Figure 2–5. Character I/O to stdout and stderr are displayed in the Console view.

Figure 2–5. Running a Program on Target Hardware

File	Edit Navigate :	Search Run Project Tools W	/indow Help			
	- 🛛 🖨 🖬] 🏇 = 🜔 = 💁 =] 🥭 /	�] �			
EP	to C/C++ Project	.s 🗙 Navigator 🛛 🗖 🔏	🖸 dhry21a.c 🕱			
C	🖃 🥰 dhrystone		<pre>#ifdef ROPT printf ("Register option selected? YES\n");</pre>			
	🗄 🚰 dhrystone	New) else			
		Go Into	<pre>printf ("Register option selected? NO\n"); strcpy(Reg_Define, "Register option not selected.");</pre>			
		Open in New Window	endif			
		Run As	Nios II Hardware			
		Debug As	Nios II Instruction Set Smulator Second: "			
		Build Project	Nios II ModelSim hrystones_Per_Second);			
		[Сору	<pre>g = %10.31f \n",Vax_Mips); printf ("\n");</pre>			
		💼 Paste	princi (\m),			
		💢 Delete	ifdef HAVE_FULLIO			
		Move	<pre>fprintf(Ap,"\n"); fprintf(Ap,"Drugtone Renghmerk, Vergion 2 1 (Lenguege);</pre>			
		Rename	<pre>fprintf(Ap, "%s\n", Reg Define);</pre>			
		🚵 Import	fprintf(Ap, "Microseconds for one loop: %7.11f\n", Microse			
		🛃 Export	fprintf(Ap,"Dhrystones per second: %10.11f\n",Dhrystones			
		🧄 Refresh				
		Close Project	id [dbrystone 0]			
		Team	ation of hardware simulation model contents and simulation			
		Compare With	 s does not affect the instruction set simulator.) 			
		Restore from Local History	t-processing to create onchip_ram_64_kbytes.hex			
		Properties	eratea 1d completed			
		200 121 22 700				

Starting a debug session is similar to starting a run session. For example, to debug the program on the ISS, right-click the project in the C/C++ **Projects** view, then click Debug As, **Nios II Instruction Set Simulator**. See Figure 2–6.

Figure 2–6. Launching the Instruction Set Simulator

E) c	/C++ - dhry2	1a.c - Nios II IDE	
File	Edit Navigate	Search Run Project Tools V	vindow Help
] 📬	- 🖪 🖻 🛛	🗟 🎋 • 🔘 • 🏊 • 🕭 .	<mark>∻ → → -]</mark> ₩ ₽
Ê	to C/C++ Proje	ects 🗙 Navigator 🛛 🗖 🛛	ⓓ dhry21a.c ⊠
Ē		수 수 🗟 🕒 😫 🕶	#ifdef ROPT
	⊕- % dhrysto ⊕-1⊖9 dhrysto	New	<pre>printf ("Register option selected? YES\n"); #else</pre>
	⊕ 🔗 hello_w	Go Into	<pre>printf ("Register option selected? NO\n");</pre>
		Open in New Window	<pre>strcpy(Reg_Define, "Register option not selected."); #endif</pre>
		Run As 🕨	printf ("Microseconds for one run through Dhrystone: ")
		Debug As 🔹 🕨	Nios II Hardware
		Build Project	Nos II Instruction Set Simulator Dhrystones_Per_Second);
		📄 Сору	<pre>printf ("VAX MIPS Fating = %10.31f \n", Vax_Mips);</pre>
		👔 Paste	<pre>printf ("\n");</pre>
		💢 Delete	#ifdef HAVE FULLIO
		Move	<pre>fprintf(Ap,"\n");</pre>
		Rename	fprintf(Ap,"Dhrystone Benchmark, Version 2.1 (Language: C
			<pre>fprintf(Ap,"%s\n",Reg_Define);</pre>
			fprintf(Ap, "Microseconds for one loop: \$7.11f\n", Microsec
		Export	Iprinti(Ap, "Durystones per Second: %10.111\n", Durystones_
		🐼 Refresh	
		Close Project	oblems 🖳 Console 🛛 Properties
		Taan	Build [dhrystone_0]
		Compare With	eation of hardware simulation model contents and simulation
		Restore from Local History	his does not affect the instruction set simulator.)
			nerated
		Properties	aild completed
		System Library Properties	
	/dhrysto	ne O	

Figure 2–7 shows a debug session in progress for the dhrystone_0 project.

Figure 2–7. Debugging dhrystone_0 on the ISS



Launching the debugger changes the workbench perspective to the debug perspective. You can easily switch between the debug perspective and the C/C++ development perspective, by clicking on the perspective icons on the left-most side of the workbench window.

After you start a debug session, the debugger loads the program, sets a breakpoint at main(), and begins executing the program. You use the usual controls to step through the code: Step Into, Step Over, Resume, Terminate, etc. To set a breakpoint, double click in the left-hand margin of the code view, or right-click and choose **Add Breakpoint**.

The Nios II IDE offers many debug views that allow you to examine the status of the processor while debugging: Variables, Expressions, Registers, Memory, etc. Figure 2–8 shows the Registers view.

Figure 2–8. The Registers View While Debugging

Tools	Window Help			
] 🙆	New Window	1 界		
	Open Perspective	र 🐐 रु रु ह 🛒 🔻		Variables Breakpoints Expressions MRegisters × Memory 🖓 🗖
on [Nic	Show View 🔸	◎ _☉ Breakpoints		約 44 🗖 🔻
2:52 P c:214	Customize Perspective Save Perspective As Reset Perspective	፼ Console 梦 Debug		── ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─
its\nios (10/1	Close Perspective Close All Perspectives	Orac Sector Control Contr		- 1000 r2 = 0xa - 1000 r3 = 0xa - 1000 r4 = 0x100b091
52 PM]	Navigation +	Memory		- 1000 r5 = 0×100b091
L	Preferences	E Outline Alt+Shift+Q, O		< >
		Registers 🖓 🖓		
(chmar	k, Version 2.1 (Lar	Trace (X)= Variables		0x01000368 <main+296>: (printf ("If you are r</main+296>
unning	g this on the Nios : several minutes to p	Other	the	following:\ 0x0100036c <main+300>: ; 0x01000370 <main+304>: ; 0x01000374 <main+308>: ;</main+308></main+304></main+300>
mance	e on the ISS will be	e optimistic compared to r	unnii	ng on hardwa printf ("1. This may

Programming Flash

Many Nios II processor systems use external flash memory to store one or more of the following items:

- Program code
- Program data
- FPGA configuration data
- File systems

The Nios II IDE provides a Flash Programmer utility to help you manage and program the contents of flash memory. Figure 2–9 shows the Flash Programmer.

Figure 2–9. The Nios II IDE Flash Programmer

rogram project to hash me	more on target hoard
Program flash with hello_world_	0 and standard.sof.
ntigurations:	Name: flash programmer
Hash Programmer	Ann Larget Connection
	Target Board: Nios Development Board, Stratix (EP1S10) Help
	V Program software project into flash memory
	Project:
	hello_world_0 Browse
	1005 H LEF Executione:
	Target Hardware
	SOPC Builder System: C:(alteral/kits/nios2)examples/verilog/niosII_stratix_1s10 standard/std_1s10.ptf Browse
	CPU: ppu 💙
	✓ Program FPGA configuration data into hardware-image region of flash memory
	Program FPGA configuration data into hardware-image region of flash memory FPGA Configuration (SOF): C1/alteral/kts/nios2lexamples/verlog/nios1I_strattx_1s10istandardistandard.sof
	Program FPGA configuration data into hardware-image region of flash memory FPGA Configuration (SOF): C1 ateral/kts/nios2 examples/verilog/niosII_stratix_1s10 standard standard.sof Browse Hardware Image: user: US + 0x600000
	Program FPGA configuration data into hardware-image region of flash memory FPGA Configuration (SOF): C1/abra1/kts/nios2/examples/verilog/nioSII_stratix_1s10\standard\stan
	Program FPGA configuration data into hardware-image region of flash memory FPGA Configuration (SOF): C1/alteral/ats/nios2/examples/verilog/niosII_stratix_1s10\standard\stan
New Delete	Program FPGA configuration data into hardware-image region of flash memory PFGA configuration (SOF): C:[alteral/kits/niss2]examples/verilog/niosII_stratx_1s10]standard/standard.sof Hardware Image: user: U5 + 0x60000 Program a file into flash memory File: Browse Memory: ext_flash (U5) Offset: 0x0

Help System

The Nios II IDE help system provides documentation on all IDE-related topics. To launch the help system, click Help, **Help Contents**. On Windows, you can also press **F1** at any time for context-sensitive help. The Nios II IDE help system contains hands-on tutorials that guides you step-by-step through the process of creating, building, and debugging a project. Figure 2–10 shows the Nios II IDE help system displaying a tutorial.

Figure 2–10. Tutorials in the Nios II IDE Help System

