Nios II Debug Client

This tutorial presents an introduction to the Nios II Debug Client, which is used to compile, assemble, download and debug programs for Altera's Nios II processor. This tutorial presents step-by-step instructions that illustrate the features of the Nios II Debug Client.

The process of dowloading and debugging a Nios II application requires the presence of an actual FPGA device to implement the Nios II processor. For the purposes of this tutorial, it is assumed that the user has access to the Altera DE2 Development and Education board connected to a computer that has Quartus II (version 5.1 or higher) and Nios II software installed.

The screen captures in the tutorial were obtained using the version 1.0 of the Nios II Debug Client; if other versions of the software are used, some of the images may be slightly different.

Contents:

Installing the Nios II Debug Client Starting the Nios II Debug Client Compiling and Downloading Nios II Applications Running an Application Using the Disassembler Examining Memory Contents Examining and Changing Register Values Single Step Advanced Single Step Setting a Breakpoint JTAG UART Console

Introduction

The Nios II Debug Client is a software application that runs on a host PC connected to a Nios II System. It allows the user to compile or assemble Nios II applications, download the application to the Nios II system and then debug the running application. The Debug Client provides functionality that allows the user to:

- Examine register and memory contents.
- Disassemble the machine code present in any memory region.
- Single step through each assembly language instruction in the Nios II application.
- Set breakpoints that trigger when certain instructions are reached or when certain data addresses are accessed.
- Examine a graphical view of an Instruction Trace that records the set of recently executed instructions before a breakpoint is hit.
- View serial output sent through the Nios II JTAG UART.

1 Installing the Nios II Debug Client



Figure 1: Install File on DE2 CD.

To install the Nios II Debug Client, proceed as follows:

- 1. Use **Microsoft Windows Explorer** to open the folder "University Program/Installs" from the DE2 CD-ROM. As shown in Figure 1, the folder will contain a single executable file named *n2client_inst*.
- 2. Double-Click on this *n2client_inst* executable file. This will bring up the first screen of the installer as illustrated in Figure 2. Click on the **Install** button and proceed to the next step. During the install, you will be asked if you would like a shortcut to the Nios II Debug Client to be placed on your **Windows Desktop**. Answering yes will install an icon similar to the one shown in Figure 3 on your desktop.
- 3. Assuming that the install was successful, the screen shown in Figure 4 will be displayed. Click on the **Finish** button to complete the installation. Should an error occur, a dialog box will suggest the appropriate action. Errors include:
 - Quartus II Software is not already installed or the Quartus II version is too old.
 - Nios II SDK Software is not already installed or the version is too old.



Figure 2: Nios II Debug Client Install Wizard.



Figure 3: Nios II Debug Client Desktop Icon.



Figure 4: Nios II Debug Client Install Finished.

2 Starting the Nios II Debug Client

Before starting the Nios II Debug Client, ensure that a Nios II System has been downloaded onto the DE2 Development and Education board. This tutorial assumes that the example Nios II system provided on the DE2 CD-ROM has been downloaded.

If you have chosen to install a Windows Desktop Shortcut, you may start the Nios II Debug Client by double clicking on the associated icon. If you have not installed the shortcut or wish to start the Debug Client in an alternative manner, proceed as follows:

 Start the *Nios II SDK Shell* from the Windows Start Menu by following the links from Altera > Nios II Development Kit > Nios II SDK Shell. This step is illustrated in Figure 5.

🖬 Altera 🕨 🕨	🖬 Nios II Development Kit 5.0	Click Here lios II IDE
	Mios II Development Kit - Evaluation Edition	5.0 Nios II SDK Shell
	🔇 Quartus II 5.0	Uninstall Location: D

Figure 5: Starting the Nios II SDK Shell.

2. The Nios II Debug Client can now be started from the Nios II SDK shell as shown in Figure 6. Enter the command *n2client* in the SDK shell and press Enter.



Figure 6: Starting the Nios II Debug Client.

After startup, the Nios II Debug Client **Settings Window** will appear as shown in Figure 7. Nios II systems have a user configurable architecture. The designer may choose from a variety of peripherals and memory options in Altera's SOPC Builder. The Nios II Debug Client needs information describing the Nios II system that is being targetted. Systems created by Altera's SOPC builder are described by a *.ptf* file located in the project's main directory. This file contains information about all the peripherals connected to the Nios II processor as well as the memory map. The Nios II Debug Client can use this information to debug applications for any Nios II based system.

*			
Nios II System Properties	k Here de2_demo	Click Here example_system	
	File <u>N</u> ame:	de2_demo.ptf	
	Files of <u>T</u> ype:	SOPC Builder PTF Files (.ptf)	Click Here
Program Memory	sdram		
Sected Memory Size : 8192	? Kbytes		
		Confirm	4

Figure 7: Nios II Debug Client Settings Window.

The **Settings Window** allows the user to choose the location of the *.ptf* file that describes their particular Nios II System. It also tells the Nios II Debug Client where it should download the target software application. This will typically be a symbolic name, created in SOPC builder, that describes a memory device. By default, the first time that the Debug Client is started, a *.ptf* file describing the example Nios II System shipped with DE2 CD-ROM will be selected and the *sdram* memory will be chosen to hold the target application. To change the defaults, proceed as follows:

- 1. Select the directory that contains your .ptf file by using the Look In: drop down list.
- 2. Double Click on the appropriate .ptf file in the SOPC Builder PTF File area.
- 3. Select the approriate memory device from the Program Memory drop down list.
- 4. Click on the **Confirm** button. These settings will remain for all subsequent runs of the Nios II Debug Client. Only the **Confirm** button will need to be pressed if no changes are required.

After a few seconds, the Nios II Debug Client window will appear. The startup screen is shown in Figure 8.

👙 Nios II Debug Client (n2client)		
File Help		
Connect Using Click Here		
USB-Blaster [USB-0]		
NIOS II Debug Output		Registers
Disassemble \ Memory \ Advanced \ Console \ Trace \ E	Debug \	Register Value
		R0 (zero) 00000000
		R2 0000000
		R3 0000000
		R5 0000000
		R6 0000000
		R7 0000000
		R8 0000000 89
		R10 0000000
		R11 00000000
		R12 0000000
		R14 00000000
		R15 0000000
		R16 0000000
		R18 00000000
		R19 0000000
		R20 0000000
		R22 00000000
		Interesting Memory
		Type Address Data
		T
Disassemble Controls		Break + Watch
Start Address (or Symbol)	Num Instructions	
Next Instruction		
NOP		
NIOSII Controls Click Here		
Compile & Load		Display Memory
Disconnect		Disassemble

Figure 8: Nios II Debug Client.

3 Compiling and Downloading Nios II Applications

To compile and download a Nios II application, proceed with the following steps:

1. As shown in Figure 8, the first step (highlighted with the circled 1) involves selecting the cable that will be used to communicate with the DE2 Development and Education board. This cable should have a name similar to *USB-Blaster [USB-0]*.

👙 Open	
Look In: Click Her	de2demo 🔹 ն 🛣 🗄 0 2
File Name:	test.s Click Here
Files of <u>T</u> ype:	Assembly Language Source Files (.s)

Figure 9: Open File Dialog Box.

- 2. Next, click on the **Compile & Load** button. A dialog box, shown in Figure 9, will appear. Select the directory that contains your application source file using the **Look In:** drop down list. Note that some sample applications are included on the DE2 CD.
- 3. For the purposes of this tutorial, we assume the existance of an assembly language source file named *test.s.* The Nios II Debug Client is capable of compiling and downloading:
 - S-Record Files: These file have a .srec extension and contain raw machine code instructions and data.
 - Assembly Language Source Files: These files have a .*s* extension and contain human readable Nios II instructions. The Nios II Debug client will call the Nios II assembler to convert these files into an equivalent S-Record format. The S-Record file will then be downloaded into the Nios II memory space.
 - C-Language Source Files: These files have a .c extension and contain C-Language code. The Nios II Debug Client will call the *gcc* compiler to compile the source code into assembly language and finally to the machine code in S-Record format. Again, the S-Record file will then be downloaded.

To select an Assembly Language source file, the user must ensure that **Assembly Language Source Files** (.s) is selected from the **Files of Type** drop down list.

- 4. Select the application to be downloaded. For this tutorial, the file *test.s* is selected.
- 5. Finally click on the **Open** button. After a few seconds, the Nios II Debug client will download the machine code into memory. The first few instructions are then disassembled and displayed as shown in Figure 10.

🖢 Nios II Debug Client (n2client)	_ 🗆 🔀
File Help	
Connect Using	
USB-Blaster [USB-0]	~
I NOS II Dobus Oxford	- Pogietore
	Desisters Habe
Disassemble Memory \Advanced \Console \Trace \Debug \	Register Value
00800000: start{C:/de2demo/test.s:15}:	R1 (at) deadbeef
00800000: orhi R2, zero, 0x0	R2 deadbeef
00800004: ori R2, R2, 0xde2	R3 deadbeef
00800008: add R3 <- zero, zero	R4 deadbeet
0080000c: add R6 <- zero, zero	Ró deadbeef
00800010; scroll(C:/de2demo/test.s:21);	R7 deadbeef 🐖
00800010: orbi R10. zero, 0x0	R8 deadbeef
00800014: ori 810, 810, 0880	R9 deadbeef
00800018: staio 82. 0x0(E10)	RIU deadbeet
ORBODIC: orbi Bil zero OxO	R12 deadbeef
00800020 ori 810 810 0.840	R13 deadbeef
	R14 deadbeef
00800012; add B12 c sero	R15 deadbeet
00800022 outer delar locu/(+/de/demo/test s:33).	R17 deadbeet
	R18 deadbeef
00000020. Dir Kr, 2200,0A1	R19 deadbeef
00000000: Immer_delay_log/c:/delaemo/cest.s:s):	R20 deadbeef
	R21 deadbeet
0000004: aud 2010 <- 2010, 2010	Interacting Mamon
0000003: bit zero d. k	The Address Date
00000000: end_inter_delay_tobp(://deldemo/cest.s:41):	Type Address Data
UGUUUSC: OFAL RIU, ZEFO, UXU	
-	
Disassemble Controls	Break + Watch
Start Address (or Symbol) 00800000 Num Instructions 10	-
- Not befraction	
	
UXUU8UUUUU: orni R2, zero, UXU	-
NIOSII Controls	
Restart Single Step Display	/ Memory
Disconnect Run Disa	ssemble

Figure 10: Source File loaded and Disassembled.

Users who wish to gain insight into details of the assembly or compilation process may use the **Debug** pane of the Nios II Debug Client, as illustrated in Figure 11. The first section of the **Debug** pane contains all of the commands used to convert a source file into machine code in the S-Record format. The user may choose to use these commands at the command prompt of the *Nios II SDK Shell* or use them as part of a *Makefile* in a more complex project.

👙 Nios II Debug Client (n2client)		
File Help		
Connect Using [US8-Blaster [US8-0] NIOS II Debug Output Disassemble \Memory \Advanced \Console \Tr Debug \ PTF: = yy=_smax = "0x00000s20"; PTF: = tag = "0x00000s20";	Register S Register Value R0 (zero) 0000000 R1 (ct) deadbeef	
<pre>PTF: isd "0x0000000000; PTF: jsd_uart "0x00000000; PTF: jtd_uart "0x00000000; PTF: jtd_uart "0x00000000; PTF: jtd_isjplay "0x00000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x00000000; PTF: vgacn_0 "0x00000000; PTF: vgacn_0 "0x00000000; PTF: vgacn_0 "0x00000000; PTF: vgacn_0 "0x00000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x00000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x00000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x0000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x00000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x0000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x00000000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x000000000; PTF: vgacn_0 "0x00000000; PTF: vgacn_0 "0x00000000; PTF: vgacn_0 "0x0000000; PTF: vgacn_0 "0x0000000; PTF: vgacn_0 "0x0000000; PTF: vgacn_0 "0x0000000; PTF: vgacn_0 "0x00000000; PTF: vgacn_0 "0x0000000; PTF: vgacn_0 "0x0000000; PTF: vgacn_0 "0x0000000; PTF: vgacn_0 "0x000000; PTF: vgacn_0 "0x000000; PTF: vgacn_0 "0x000000; PTF: vgacn_0 "0x000000; PTF: vgacn_0 "0x000000; PTF: vgacn_0 "0x00000; PTF: vgacn_0 "0x00000; PTF: vgacn_0 "0x00000; PTF: vgacn_0 "0x00000; PTF: vgacn_0 "0x00000; PTF: vgacn_0 "0x00000; PTF: vgacn_0 "0x0000; PTF: vgacn_0 "0x0000; PTF: vgacn_0 "0x0000; PTF: vgacn_0 "0x0000; PTF: vgacn_0 "0x0000; PTF: vgacn_0 "0x0000; PTF: vgacn_0 "0x000; PTF: vgacn_0 "0x00; PTF: vgacn_0 "0x00; PTF: v</pre>	R2 deadbeef R3 deadbeef R4 deadbeef R5 deadbeef R6 deadbeef R7 deadbeef R9 deadbeef R10 deadbeef R11 deadbeef R12 deadbeef R13 deadbeef R14 deadbeef	
<pre>=== Linking C:/de2demo/test.s.o d/altera new/kits/nios2/bin/nios2-gnutools/H-i686-pc-cygwin/bin/nios2-elf-ld.exe -Ttext=00800000 -e it 4 u _start -g -o C:/de2demo/test.elf C:/de2demo/test.s.o === ne Linking === Creating SREC C:/de2demo/test.srec d:/altera_new/kits/nios2/bin/nios2-gnutools/H-i686-pc-cygwin/bin/nios2-elf-objcopy.exe -0 srec C:/de2demo/test.elf C:/de2demo/test.srec</pre>		
Next Instruction		
0x00800000: orhi R2, zero, 0x0		
NIOSII Controls		
Restart Single Step D	Display Memory	
Disconnect Run	Disassemble	

Figure 11: Debug Information.

The **Debug** pane will also be populated with other types of advanced information as the debugging session progresses. The Nios II Debug Client is a graphical interface that communicates with the *nios2-gdb-server* provided with *Altera's Nios II SDK*. The *nios2-gdb-server* provides a set of low-level routines that allow for debugging an application running on a Nios II system. A description of the commands and protocol for communicating to the server is described in http://sourceware.org/gdb/current/onlinedocs/gdb_33.html# SEC684. The **Debug** pane displays each of these low level commands and responses received for every action performed in the Debug Client.

4 Running an Application

To run the target application on the DE2 board, execute the following steps:

1. Return to the **Disassemble** pane and Click the **Run** button as shown in Figure 10.

Source State (Street (n2client)		_ 🗆 🗙
File Help		
Connect Using		
US8-Blaster [US8-0]		~
NIOS II Debug Output	Registers	
Disessemble) Mamory / Edvarced / Corocle / Trace / Debug	Register	Value
Essassemble (Mellio) y (Advanced (Console (Trace (Debug (RO (zero)	00000000 🔺
008800000: _start(C:/de2demo/test.s:15):	R1 (at)	deadbeef
UBSUDUCU: orhi R2, zero, UXU	R3	deadbeet
UBSUUU4: ori k2, k2, Uxde2	R4	deadbeef
UBSUUUS: add k3 <- zero, zero	R5	deadbeef
	RD P7	deadbeet
	R8	deadbeef
	R9	deadbeef
	R10	deadbeef
OBSOUCH: other Bio, sero, OXO	R12	deadbeef
00800020: ori 810, 810, 0x840	R13	deadbeef
00800024: stwip 86, 0x0(810)	R14	deadbeef
00800028: add R12 <- zero, zero	R15	deadbeet deadbeef
0080002c: outer delay loop(C:/de2demo/test.s:33):	R17	deadbeef
0080002c: ori R7, zero, 0x1	R18	deadbeef
00300030: inner delay loop(C:/de2demo/test.s:35):	R19	deadbeet deadbeet
00800030: addi R7, R7, -0x1	R21	deadbeef
00800034: add zero <- zero, zero	R22	deadbeef 💌
00800038: bne zero, R7, -0xc @inner_delay_loop{C:/de2demo/test.s:35}	_Interesting Me	emory
0080003c: end_inner_delay_loop{C:/de2demo/test.s:41}:	Type Ac	idress Data
0080003c: orhi R10, sero, 0x0		
Disassemble Controls	Break + Watc	:h
Start Address (or Symbol) 00800000 Num Instructions 10		r
Next Instruction		-
0x00800000: orhi R2, zero, 0x0		
NIOSII Controls		
Restart Single Sten		
Disconnect Hait Disa		

Figure 12: Program Running.

2. While the application is running, the Nios II Debug Client will only have a single button that is enabled. This is the **Halt** button and is shown in Figure 12. This button can be used to stop the execution of the target application and allow for debugging.

👙 Nios II Debug Client (n2client)		
File Help		
Connect Using		
		-
NOS II Debug Output	Registers	
Disassemble Mamory) Edvarpad) Consolo) Traco) Dahura)	Register	Value
Disassemble (Metholy (Advanced (Colisole (Thace (Debug))	RO (zero)	00000000 🔺
00800000: _start(C:/de2demo/test.s:15):	R1 (at)	deadbeef
00800000: orhi R2, zero, 0x0	RZ	UUde2UUU
00800004: ori R2, R2, 0xde2	R4	deadbeef
00800008: add R3 <- zero, zero	R5	00000000
0080000c: add R6 <- zero, zero	Ró	11975225
00800010: scroll{C:/de2demo/test.s:21}:	R7	00000000
00800010: orhi R10, zero, 0x0	PQ R0	deadbeef
00800014: ori R10, R10, 0x880	R10	00000840
00800018: stwic R2, 0x0(R10)	R11	deadbeef
0080001c: orhi R10, zero, 0x0	R12	00000000
00800020: ori R10, R10, 0x840	R13 D14	deadheef
00800024: stwio R6, 0x0(R10)	R15	deadbeef
00800028: add R12 <- zero, zero	R16	deadbeef
0080002c: outer_delay_loop{C:/de2demo/test.s:33	R17	deadbeef
0080002c: ori R7, zero, 0x1	D10	deadbeet
00800030: inner_delay_loop(C:/de2demo/test.s:35	R20	deadbeef
00800030: addi R7, R7, -0x1	R21	deadbeef
00800034: add zero <- zero, zero	R22	deadbeef 💌
00800038: bne zero, R7, -Oxc @inner_delay_loop(C:/de2demo/test.s:35)	Interesting M	lemory
0080003c: end_inner_delay_loop(C:/de2demo/test.s:41):	Type A	ddress Data
0080003c: orhi R10, zero, 0x0		
Disassemble Controls	Break + Wat	ch
Start Address (or Symbol) 00800000 Num Instructions 10		-
		-
		v
0x0080002c: ori R7, zero, 0x1		·
NIOSII Controls		
Restart Single Step Display	Memory	
Click Here		
Disconnect Disas		

Figure 13: Program Stopped.

3. Click on the **Halt** button as shown in Figure 13. The Nios II Debug Client will then pop up a dialog box indicating that the application has been halted. Press the **OK** button and proceed to the next step.

Sios II Debug Client (n2client)	
File Help	
Connect Using	
USB-Blaster [USB-0]	•
NIOS II Debug Output	Ponistors
	Degister Value
Disassemble \Memory \Advanced \Console \Verace \Debug \	R0 (zero) 00000000
$A_B_C_D_E$	R1 (at) deadbeef
	R2 00de2000
ABCDE	R3 00000001
	R5 00000000
A:	R6 11975225
00800010; scroll{C:/de2demo/test.s:21};	R7 00000000
	R8 00000000 💥
00800014. ori 110 110 02880	R10 00000840
0000019, $0110000000000000000000000000000000000$	R11 deadbeef
0020001d: Stwite R2, 0X0(R10)	R12 00000000
	R13 00000000 P14 deadbeef
	R15 deadbeef
00800024: stwie R6, 0x0(R10)	R16 deadbeef
00800028: add R12 <- zero, zero	R17 deadbeef
0080002c: outer_delay_Loop{C:/de2demo/test.s:33}:	R18 deadbeet
0080002c: ori R7, zero, 0x1	R20 deadbeef
00800030: inner_delay_loop{C:/de2demo/test.s:35}:	R21 deadbeef
00800030: addi R7, R7, -0x1	R22 deadbeef
00800034: add zero <- zero, zero	Interesting Memory
00800038: bne zero, R7, -0xc @inner_delay_loop{C:/de2demo/test.s:35	Type Address Data
Y	
B	Break + Watch
0000002g, and inpan dalay loon (C:/da?damo/tast c:/11.	-
Next Instruction	-
0x0080002c: ori R7, zero, 0x1	_
NIOSII Controls	
Restart Single Stan Displa	v Memory
	y nomory
Disconnect Run Disa	assemble

Figure 14: Instruction Trace.

- 4. Once the application has been halted, the user can view an instruction trace that shows the recently executed instructions just before the point at which the halt occured. This is accomplished by clicking on the **Trace** tab as shown in Figure 14.
- 5. Notice also that the values of the various Nios II registers have been updated in the **Registers** table. These values contain the register contents at the halt point.

5 Using the Disassembler

To disassemble the contents from a particular set of memory locations, proceed as follows:

👙 Nios II Debug Client (n2client)		
File Help		
Connect Using		
USB-Blaster [USB-0]		
Nos Click Here	Dogietore	
	Desisters	Unive
Disassembly ory \Advanced \Console \Trace \Debug \	Register PO (zero) DOC	
	R1 (at) dea	ubeef 📃
00800010: scroll(C:/de2demo/test.s:21):	R2 dea	dbeef
00800010: orhi R10, zero, 0x0	R3 dea	dbeef
00800014: ori R10, R10, 0x880	R4 dea D5 dea	dbeet
00800018: stwic R2, 0x0(R10)	Ró dea	dbeef
0080001c: orhi R10, zero, 0x0	R7 dea	dbeef 😹
00800020: ori R10, R10, 0x840	R8 dea	dbeef 🕺
00800024: stwio R6, 0x0(R10)	RY dea D10 dea	dbeet dbeef
00800028: add R12 <- zero, zero	R10 dea	dheef
0080002c: outer_delay_loop(C:/de2demo/test.s:33):	R12 dea	dbeef
0080002c: ori R7, zero, 0x1	R13 dea	dbeef
00800030: inner_delay_loop(C:/de2demo/test.s:35):	R14 dea	dbeef
00800030: addi R7, R7, -0x1	R15 dea D16 dea	dbeet
00800034: add zero <- zero, zero	R17 dea	dbeef
00800038: bne zero, R7, -Oxc @inner delay loop(C:/de2demo/test.s:35)	R18 dea	dbeef
0080003c: end inner delay loop{C:/de2demo/test.s:41}:	R19 dea	dbeef
0080003c: orhi RIO, zero, 0x0	R20 dea	dbeef
00800040: ori R10, R10, 0x850	RZI dea P22 dea	dbeet 🗸
00800044: 1dwio R8, 0x0(R10)	Interacting Many	
00800048: addi R12, R12, 0x1	interesting werni	UIY
0080004c: cmpge R13 <- R8, R12	Type Addre	ss Data
Disassemble Controls Click Here Click Here	Break + Watch-	
Start Address (or Symbol) scroll Num Instructions 16	•	
	-	
Next Instruction	•	
0x00800000: orhi R2, zero, 0x0	•	
NIOSII Controls		
Restart Single Step Display I	Memory	
Click Here	•	
Disconnect Run Disass	emble	

Figure 15: Using the Disassembler.

- 1. Click on the **Disassemble** tab as shown in Figure 16.
- 2. Enter the starting address of the memory block to be displayed as a HEX value (there is no need to use a prefix such as "0x") in the **Start Address** text field. The user may also choose to enter a symbolic name which was present in the original source file. The Nios II Debug Client will map this name to an address and use this value as the starting address. If the symbolic name is not found, an error will be issued.
- 3. Enter the number of instructions to be disassembled in the **Num Instructions** text field. This is a decimal value. Note that the maximum value is limited to 64 instructions.
- 4. Click on the **Disassemble** button and the disassembled source code will appear in the text area. Note that the disassembler will display both your original source code and the disassembled instructions if possible. This is typically possible for files that were assembled or compiled. It is not possible for manually created S-Records that contain only machine code.
- 5. If desired, a printed copy of the contents of the text area can be produced by clicking on the **Print** button.

6 Examining Memory Contents

To examine the contents of the Nios II memory space, execute the following steps:

👙 Nios II Debug Client (n2client)			
File Help			
Connect Using			
USB-Blaster [USB-0]			*
			Registers
	Dahua		Register Value
Disassemble wemory (Advanced (Console (Trace)	Debug \		R0 (zero) 00000000 🔺
			R1 (at) deadbeef
00800000: 00800034 10837894			P3 00000001
00800008: 0007883a 000d883a			R4 deadbeef
00800010: 02800034 52822014		88	R5 00000000
00800018: 50800035 02800034			R6 11975225
00800020: 52821014 51800035			R8 00000000
00800028: 0019883a 01c00054			R9 deadbeef
00800030: 39ffffc4 0001883a			R10 00000840
00800038: 383ffdle 02800034			R12 00000000
00800040: 52821414 52000037			R13 00000000
00800048: 63000044 43184038			R14 deadbeef
00000050, 10044120, 00000106			R15 deadbeet
00800060 1004013a 114003~~			R17 deadbeef
00800068: 28000226 0007883a			R18 deadbeef
00800070: 00000306 117c002c			R19 deadbeet
00800078: 28000126 00c00044		•	R21 deadbeef
-Memory Controls			R22 deadbeef
Click Here			Interesting Memory
Start Address 800000		Mem Format	Type Address Data
Clic	k Here	Bytes Click Here	· · · · · · · · · · · · · · · · · · ·
End Address 800100		16-Bit Words	
	43		
Eil Value		Mirita to Mamory	Break + Watch
		vance comeniory	-
-Next Instruction			-
			-
0x0080002c: ori R7, zero, 0x1			
-NIOSII Controls		Click Here	
Restart	Single Step	Display	/ Memory
			, riomory
Disconnect	Bup	Disa	ssemble
Disconnece	Nurt	Disa	330mbio

Figure 16: Examining Memory.

- 1. Click on the Memory tab as shown in Figure 16.
- 2. Enter the starting address of the memory block to be displayed as a HEX value (there is no need to use a prefix such as "0x") in the **Start Address** text field.
- 3. Enter the ending address of the memory block to be displayed as a HEX value (again, there is no need to use a prefix such as "0x") in the **End Address** text field.
- 4. Select the format of the memory display from the **Mem Format** list. In Figure 16, the value **32-bit Words** is selected.
- 5. Finally, click on the **Display Memory** button. The Nios II Debug client will respond by fetching the requested data and displaying it in the Memory text area.

7 Examining and Changing Register Values

The process of changing a register value is as follows:

👙 Nios II Debug Client (n2client)				
File Help				
Connect Using				
			~	
NIOS II Debug Output			Registers	
Disassemble Memory Advanced Console Trace	Debug \		Re	
	Dibbilg (R0 (ze Double Click	
0000000 0000001 10007004			R2 Enter New Hex	
			R3 Value	
008000000: 00070034 00000034			R4	
00800018: 50800035 02800034			R6 11 5225	
00800020: 52821014 51800035			R7 00000000	
00800028: 0019883a 01c00054			R8 0000000 R8 deadbeef	
00800030: 39ffffc4 0001883a			R10 00000840	
00800038: 383ffd1e 02800034			R11 deadbeef	
00800040: 52821414 52000037			R12 00000000 R13 00000000	
00800048: 63000044 431a403a			R14 deadbeef	
00800050: 683ff61e 18000226			R15 deadbeef	
00800058: 1004d13a 00000106			R10 deadbeet	
00800060: 1004913a 114003cc			R18 deadbeef	
00800070: 00000306 117~002~			R19 deadbeef	
00800070: 00000306 11700020		-	R21 deadbeef	
-Momory Controls			R22 deadbeef	
			Interesting Memory	
Start Address 800000		Mem Format	Type Address Data	
		Bytes		
End Address 800100		32-Bit Words		
Fill Value		Write to Memory	Break + Watch	
				
Next Instruction				
0x0080002c: ori R7, zero, 0x1			•	
NIOSII Controls				
Restart	Single Step	Displa	y Memory	
Disconnect	Run	Disa	issemble	
00800020: 52821014 51800035 00800028: 0019883a 01c00054 00800030: 39ffffc4 0001883a 00800034 00800048: 383ffd1e 02800037 00800048: 63000044 431a403a 00800050: 683ff61e 00800050: 683ff61e 18000226 00800058: 1004d13a 0000106 00800058: 28000226 007883a 00800070: 0000306 117c002c 00800078: 28000126 00800078: 28000126 00c00044 Memory Controls Start Address Start Address 800000 Fill Value	Single Step	Mem Format Bytes 16-8it Words 52-8it Words Write to Memory	R8 000000000 R9 deadbeef R10 00000000 R11 deadbeef R12 00000000 R13 00000000 R14 deadbeef R15 deadbeef R16 deadbeef R17 deadbeef R18 deadbeef R20 deadbeef R21 deadbeef R22 deadbeef R21 deadbeef R22 deadbeef R21 deadbeef R22 deadbeef V Type Address Data V V V V V V V V V V V V V V V V V V V V V V V V V V V V V V <t< td=""></t<>	

Figure 17: Changing a Register Value.

 Changing the value in register R7 is illustrated in Figure 17. Double click on the text field associated with R7 and enter a new HEX value. Press the Enter key to complete the change.

🖕 Nios II Debug Client (n2client)				
File Help				
Connect Using				
			*	
NIOS II Debug Output			Registers	
Disassemble Memory Advanced Console Trace [Debua \		Register Value	
	- 02 dg {		R0 (zero) 00000000	
			RI (at) deadbeet R2 00de2000	
			R3 00000001	
			R4 deadbeef	
00800018: 50800035 02800034			R6 11975225	
00800020: 52821014 51800035			R7 abcdef12	
00800028: 0019883a 01c00054			R8 00000000	
00800030: 39ffffc4 0001883a			R10 00000840	
00800038: 383ffd1e 02800034			R11 deadbeef	
00800040: 52821414 52000037			R12 00000000	
00800048: 63000044 431a403a			R14 deadbeef	
00800050: 683ff61e 18000226			R15 deadbeef	
00800058: 1004d13a 00000106			R16 deadbeet	
00800060: 1004913a 114003cc			R18 deadbeef	
00800068: 28000226 0007883a			R19 deadbeef	
			R20 deadbeet	
00800078: 28000126 00200044		_	R22 deadbeef	
Memory Controls			Interesting Memory	
Start Address 800000		Mem Format	Type Address Data	
		Bytes		
End Address 800100		16-Bit Words		
		32-bit words		
cil value		Mirito to Momoru	Break + Watch	
		while to Memory	-	
Next notruction			-	
				
0x0080002c: ori R7, zero, 0x1				
NOSII Controls				
Restart	Single Step	Displa	y Memory	
Disconnect	Run	Disa	issemble	

Figure 18: Register Value Changed.

2. Figure 18 shows the result of changing the contents of register **R7** to the HEX value **abcdef123**.

Although, only a change of **R7** has been described, all registers (including the program counter) can be changed in the same manner.

8 Single Step

The Nios II Debug Client makes it possible to single step through a downloaded program. Each step consists of executing a single assembly language instruction and returing control to the Nios II Debug Client. If the user has chosen to compile a C-program, the single steps will still correspond to assembly language generated from the C-code. The functionality of stepping through entire statements in the high level source is not supported by the Nios II Debug Client; however, the *Nios II Integrated Development Environment* supports these advanced features.

Notice in Figure 18 that the Nios II Debug Client indicates that the **Next Instruction** to be executed is one which will load the register **R7** with the value **1**. This will be different in your case depending on the application downloaded and where it was stopped; however, the general technique of the single step will remain the same.

👙 Nios II Debug Client (n2client)			_ 🗆 🔀
File Help			
Connect Using			
USB-Blaster [USB-0]			~
NIOS II Debug Output			Registers
(Disassamble) Memory) Eduanced) Console (Trace)	Dobug)		Register Value
Disassemble Memory Advanced (Console (mace)	Debug /		R0 (zero) 00000000 🔺
		_	R1 (at) deadbeef
00800000: 00800034 10837894			R2 UUde2UUU R3 00000001
00800008: 0007883a 000d883a			R4 deadbeef
00800010: 02800034 52822014		202	R5 0000000
00800018: 50800035 02800034			R6 11975225
00800020: 52821014 51800035			
00800028: 0019883a 01c00054			R9 deadbeef
00800030: 39ffffc4 0001883a			R10 00000840
00800038: 383ffd1e 02800034			R11 deadbeef
00800040: 52821414 52000037			R12 00000000
00800048: 63000044 431a403a			R14 deadbeef
00800050: 683ff61e 18000226			R15 deadbeef
00800058: 1004d13a 00000106			R16 deadbeef
00800060: 1004913a 114003cc			R17 deadbeef
00800068: 28000226 0007883a			R18 deadbeet
00800070: 00000306 117c002c		R20 deadbeef	
00800078: 28000126 00c00044		-	R21 deadbeef
Memory Controls			R22deadbeef
			Interesting Memory
Start Address 800000		Mem Format	Type Address Data
		Bytes	
End Address 800100		16-Bit Words	
		SE DIC WORDS	
Fill Volue		Mirita to Mamory	Break + Watch
		write to Memory	
-Next Instruction			
0x00800030 addi B7 B7 -0x1			
	Click Hara		
Restart	Single Step	Display	/ Memory
Disconnect	Run	Disas	ssemble

Figure 19: Single Step.

Click on the **Single Step** button as shown in Figure 19. After execution of this single instruction, the Nios II Debug Client will update the register values of any affected registers. In this example, we expect $\mathbf{R7} = 1$ and this is indeed the case.

9 Advanced Single Step

Often, a user may wish to single step through an application until some set of conditions is satisfied. In this example, we illustrate how to single step through the application until the condition R2=DE2000 is met. The

🐇 Nios II Debug Client (n2client)					
File Help					
Connect Using					
USB-Blaster [USB-0]					
NIOS II Debug Outp Click Here	Registers				
	Disassemble \ Memory \ A() ced \ Console \ Trace \ Debug \				
Expression Click Here		R0 (zero) 00000000			
R2 == Hde20000		R2 00de2000			
$\sqrt{2}$		R3 0000001			
		R5 0000000			
		R6 11975225			
		R7 00000001			
		R9 deadbeef			
		R10 0000840			
		RII deadbeet			
		R13 00000000			
		R14 deadbeef			
		RID deadbeet			
		R17 deadbeef			
		R18 deadbeef			
		R19 deadbeet			
		R21 deadbeef			
		R22deadbeef			
		Interesting Memory			
		Type Address Data			
Devil					
Click Here		-Brook - Wateh			
Single	Stan until (avaraccion trua)				
<u> </u>	s step tilliti (expression == tilde)				
Next Instruction					
0x00800030: addi R7, R7, -0x1					
NOSII Controls					
Restart	Single Step	Display Memory			
Disconnect	Run	Disassemble			

Figure 20: Advanced Tab.

process is illustrated in Figure 20:

- 1. Click on the **Advanced** tab.
- 2. Enter the condition to be searched for in the **Expression** window. Notice that HEX values require the prefix "**H**".
- 3. Click on the **Single step until** button to single step through the program until the condition specified in the expression expression is met.

4. Once the condition has been met, the Nios II Debug Client will stop executing single instructions and return control to the user as shown in Figure 21. Notice that the value of **R2** is as expected in the **Registers** table.

👙 Nios II Debug Client (n2client)			
File Help			
Connect Using			
USB-Blaster [USB-0]			~
NIOS II Debug Output			Registers
			Register Value
Disassemble (Memory Advanced (Console Trace	Debug (R0 (zero) 00000000
Expression			R1 (at) deadbeef
R2 == Hde20000			R2 0de20000
			R3 deadbeef
			R5 00000000
			R6 1197522b
			R7 00000000
			R8 UUUUUUUU 🛞
			R10 00000850
			R11 deadbeef
			R12 00000001
			R13 00000000
			R15 deadbeef
			R16 deadbeef
			R17 deadbeef
			R18 deadbeef
			R19 deadbeet
			R21 deadbeef
			R22 deadbeef 💌
			Interesting Memory
			Type Address Data
63	10-10-10-10-10-10-10-10-10-10-10-10-10-1		
Result			
R2=0de20000 : 1			Break + Watch
Single Step until (expression == true)			
- Next Instruction			
0x00800064. andi B5 B2 0xf			
NO3I COROS			
Restart	Single Step	Display	Memory
Disconnect	Pup	Disconamble	
Disconnect	Kuit	Disas	semple

Figure 21: Single Step Until Condition Met.

More complex expression formulations may be used as C-language style logical expressions are supported. For example, to check for the condition where registers **R1**, **R2** and **R3** all contain the same value the expression would be: R1==R2 & R2==R3.

10 Setting a breakpoint

Breakpoints are special conditions that are checked by dedicated hardware in the Nios II processor as the application is running in real time. The difference between a breakpoint and the **Advanced Single Step** is that breakpoints only support very limited conditions to be checked for such as:

- An instruction at a particular address has been reached.
- A particular address has been read from.
- A particular address has been written to.
- A particular address has been accessed.

Although the conditions are limited, they are checked in real time as the application is executed.

👙 Nios II Debug Client (n2client)				
File Help				
Connect Using				
USB-Blaster [USB-0]			~	
NIOS II Click Here		Regi	sters	
Disassemble Memory Advanced Console Trace (Debug)	Re	agister Value	
	Debug (RO (z	.ero) 00000000 🔺	
00800004: C. RZ, 0XdEZ		R1 (a	t) deadbeef	
00800000: add $R5 < zero, zero$		R3	00000000	
00800000: add ko (2010, 2010 00800010: scroll{C:/de2demo/test.s:21}:		R4	deadbeef	
00800010: orhi R10, zero, 0x0		R5	00000000 1107522b	
00800014: ori R10, R10, 0x880		R7	00000000 220	
00800018: stwic R2, 0x0(R10)		R8	00000000	
0080001c: orhi R10, zero, 0x0		R9	deadbeef	
00800020: ori R10, R10, 0x840		R10	deadbeef	
00800024: stwio R6, 0x0(R10)		R12	00000001	
00800028: add R12 <- sero, sero		R13	00000000	
0080002c: outer_delay_loop{C:/de2demo/test	:.s:33}:	R14	deadbeet	
0080002c: ori R7, zero, 0x1		R16	deadbeef	
00800030: inner_delay_loop{C:/de2demo/test	:.s:35}:	R17	deadbeef	
00800030: addi R7, R7, -0x1		R18	deadbeet	
00800034: add zero <- zero, zero		R17	deadbeef	
00800038: bne sero, R7, -0xc @inner_delay_loop(C:/de2demo/test.s:35)			deadbeef	
0080003c: end_inner_delay_loop{C:/de2demo/	(test.s:41):	IR22	deadbeef	
0080003c: orhi R10, zero, 0x0		Inter	esting Memory	
		Ty	pe Address Data	
		~		
	ž			
Disassemble Controls		Click	K Here ch Click Here	
Start Address (or Symbol) 00800000	Num Instructions 10	break	3 800020 (2)	
Next Instruction			-	
0x00800064: andi R5, R2, 0xf				
NIOSII Controls				
Restart	Click Here Single Step	Display Memory		
Disconnect	Run	Disassemb	le	

Figure 22: Setting a breakpoint.

This tutorial will demonstrate setting a breakpoint based on a certain instruction being reached. This is illustrated in Figure 22. The process of setting the breakpoint is as follows:

1. First click on the **Disassemble** tab to determine the instruction at which the breakpoint should be set. In this example, we choose the instruction at *outer_delay_loop:* and determine from the disassembly view that it is located at address **80002c**.

- 2. Click on the first text field in the Break & Watch section. Enter the address 80002c.
- 3. Click on the drop down list next to the address just entered. Select **break**. This corresponds to a breakpoint when a particular instruction has been reached. Other choices include **rwatch** (address read from), **wwatch** (address written to) and **awatch** (address accessed either by a read or write).
- 4. The next step involes running the program by clicking on the **Run** button.
- 5. When the target instruction has been reached, the Nios II Debug Client will indicate this condition as shown in Figure 23. Click **OK** to continue debugging.

👙 Nios II Debug Client (n2client)					
File Help					
Connect Using					
					-
			-Dovictore -		
NIOS II Debug Output			Registers		
Disassemble \Memory \Advanced \Console \Trace \I	Debug \		Register	Value	
00800004: ori 82, 82, 0xde2			RU (zero)	00000000	
00800008: add P3 <- zero zero			P2	0.4e 20000	_
00800000; add R6 (= pero, pero			R3	00000000	
000000000: add R0 <- 200, 200			R4	deadbeef	
UU8UUUIU: Scroll{C:/dezdemo/test.s:21}:			R5	00000000	
UU8UUUIU: orhi RIU, zero, UXU			Ró	1197522c	_
00800014: ori R10, R10, 0x880			R7	00000000	
00800018: stwio R2, 0x0(R10)			R8	deadboof	- 28
0080001c: orhi R10, zero, 0x0			R10	00000840	_
00800020: ori R10, R10, 0x840			R11	deadbeef	
00800024: stwio R6, 0x0(R10)			R12	00000000	
00800028: add R12 <- zero, zero 🦟	(R13	00000000	
0080002c: outer delay loop{C:/de2demo/te	essage	×	R14	deadbeef	_
0080002c: ori 87. zero, 0x1	•		R15	deadbeet	
00800030: inner deleu loon/C:/de2demo/te	Program Halted because of hardware breakpoint at: 0080003	2c	R10 D17	deadbeet	
000000000. Inner_deray_100p(0.)desdemo/de	Click Here		R18	deadbeef	
OUG000030: addi R/, R/, -Ux1			R19	deadbeef	
00800034: add zero <- zero, zero			R20	deadbeef	
00800038: bne zero, R7, -0xc @inner_delay_loop(c:/deldemo/lesc.s:ss)			R21	deadbeef	
0080003c: end_inner_delay_loop{C:/de2demo/	test.s:41}:		RZZ	deadbeet	
0080003c: orhi R10, zero, 0x0			Interesting	Memory	
			Туре	Address D/	ata
Disassemble Controls			-Break + Wa	tch	
Start Address (or Symbol) 00800000	Num Instructions 10			▼ 80002c	
				-	
Next Instruction				•	_
0x0080002c: ori R7, zero, 0x1				-	
NIOSII Controls					
Restart					
Disconnect	Halt				

Figure 23: Breakpoint hit.

11 JTAG UART Console

For applications that send text data through the Nios II JTAG UART (such a the *printf* routine in "C"), the Nios II Debug Client is capable of displaying this information. Click on the **Console** tab to view any output sent by the JTAG UART. Text input (i.e. *scanf*) is not supported at this time.

👙 Nios II Debug Client (n2client)			
File Help			
Connect Using			
USB-Blaster [USB-0]			~
NIOS II Debug Output Click Here		Regist	ers
			ister Value
Disassemble (Memory (Advanced grisole (Trace	Debug (RO (zei	°o) 00000000 🔺
JTAG UART link established using cable "USB-Blaster [USB-0]", device 1, instance 0x00			deadbeef
Sending Code 81 (Right Arm Up)			deadbeef
Sending Code 84 (Right Arm Down)		R5 P4	deadbeet
Sending Code 89 {Left Arm Up}		R5	deadbeef
Sending Code 8c {Left Arm Down}		R6	deadbeef
Sending Code 83 {Tilt Body Right}		R7	deadbeef
Sending Code 8b (Tilt Body Left)			deadbeet
Sending Code 8b {Tilt Body Left}		R10	deadbeef
		R11	deadbeef
		R12	deadbeef
		R13	deadbeet
		R14 P15	deadbeet
		R16	deadbeef
		R17	deadbeef
		R18	deadbeef
		R19	deadbeef
		R20	deadbeet
		rinteres	stina Memory
		Type	Address Data
		Break	+ Watch
			-
Next Instruction			-
Quello anti an zono Quil	0.0		-
UXUU8UUUUU: orni sp, zero, UXIUU			
NIOSII Controls			
Restart		Display Memo	
Discussed			
Disconnect	Hait		

Figure 24: Console Tab.

Copyright ©2006 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, mask work rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

This document is being provided on an "as-is" basis and as an accommodation and therefore all warranties, representations or guarantees of any kind (whether express, implied or statutory) including, without limitation, warranties of merchantability, non-infringement, or fitness for a particular purpose, are specifically disclaimed.