Chapter 1. My First Nios II Software Design
Software and Hardware Requirements ................................................................. 1–1
Download Hardware Design to Target FPGA ......................................................... 1–3
Nios II SBT for Eclipse Build Flow ................................................................. 1–5
  Create the Hello World Example Project ......................................................... 1–5
  Build and Run the Program .............................................................................. 1–9
  Edit and Re-Run the Program .......................................................................... 1–10
  Debugging the Application .............................................................................. 1–11
  Why the LED Blinks .......................................................................................... 1–12
Board Support Package ....................................................................................... 1–12
Next Steps .......................................................................................................... 1–15

Additional Information
Document Revision History .................................................................................. Info–1
How to Contact Altera .......................................................................................... Info–1
Typographic Conventions ..................................................................................... Info–1
1. My First Nios II Software Design

This tutorial provides comprehensive information to help you understand how to create a software project for a Nios II processor system in an Altera FPGA and run the software project on your development board.

The Nios® II processor core is a soft-core CPU that you download (along with other hardware components that comprise the Nios II system) onto an Altera FPGA. This tutorial introduces you to the basic software development flow for the Nios II processor. In the tutorial, you use a simple, pre-generated Nios II hardware system and create a software program to run on it.

The example Nios II hardware system provides the following necessary components:

- Nios II processor core
- Off-chip DDR memory interface to store and run the software
- USB serial link for communication between the host computer and the target hardware (typically using a USB-Blaster™ cable)
- LED peripheral I/O (PIO)

Software and Hardware Requirements

This section assumes you have already installed the Quartus® II design software, the Nios II Embedded Design Suite, and your development kit software. Figure 1–1 shows an example of the installation directory structure.
The tutorial describes how to use the Nios II tools with different development kits. Table 1–1 describes the target hardware design files and location for development kits the tutorial supports.

Table 1–1. Development Kit Information

<table>
<thead>
<tr>
<th>Kit</th>
<th>Design Type</th>
<th>File / Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclone III Development Kit (Cyclone III 3C120)</td>
<td>Design files directory</td>
<td>Download the design example from the Standard Nios II Hardware Design Example page of the Altera website.</td>
</tr>
<tr>
<td>Cyclone III Development Kit (Cyclone III 3C120)</td>
<td>SRAM Object File</td>
<td>cycloneII_3c25_niosII_standard.sof</td>
</tr>
<tr>
<td>Cyclone III Development Kit (Cyclone III 3C120)</td>
<td>SOPC Information File</td>
<td>cycloneII_3c25_niosII_standard_sopc.sopcinfo</td>
</tr>
<tr>
<td>Cyclone III Development Kit (Cyclone III 3C120)</td>
<td>SRAM Object File</td>
<td>tserd_3c120.sof</td>
</tr>
<tr>
<td>Cyclone III Development Kit (Cyclone III 3C120)</td>
<td>SOPC Information File</td>
<td>tserd_3c120_sopc.sopcinfo</td>
</tr>
</tbody>
</table>
Before starting the tutorial, copy the design files directory described in Table 1–1 for your kit to the location where you plan to run the tutorial. Throughout the tutorial, <your project directory> refers to this directory.

**Download Hardware Design to Target FPGA**

The software that you build will be executed by a Nios II processor-based system in an FPGA. Therefore, the first step is to configure the FPGA on your development board with the pre-generated Nios II standard hardware system. Download the FPGA configuration file, that is, the SRAM Object File (.sof) that contains the Nios II standard system, to the board by performing the following steps:

1. Connect the board to the host computer using the USB download cable.
2. Apply power to the board.
4. Accept the default workspace. We will be changing this later.
5. On the Nios II Tools menu, click Quartus II Programmer.
6. Click Hardware Setup in the upper left corner of the Quartus II Programmer window. The Hardware Setup dialog box appears.
7. Select USB-Blaster from the Currently selected hardware list, as shown in Figure 1–2.

If the download cable does not appear in the list, you must first install drivers for the cable. For information about download cables and drivers, refer to the Download Cables page of the Altera website.

---

**Table 1–1. Development Kit Information**

<table>
<thead>
<tr>
<th>Kit</th>
<th>File Type</th>
<th>File / Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratix IV GX FPGA Development Kit (Stratix IV GX 4SGX230)</td>
<td>Design files directory</td>
<td><code>&lt;Nios II EDS installation directory&gt;</code>\examples\verilog\niosII_stratixIV_4sgx230\triple_speed_ethernet_design directory (1)</td>
</tr>
<tr>
<td></td>
<td>SRAM Object File</td>
<td><code>tserd_4sgx230.sof</code></td>
</tr>
<tr>
<td></td>
<td>SOPC Information File</td>
<td><code>tserd_4sgx230_sopc.sopcinfo</code></td>
</tr>
</tbody>
</table>

**Notes to Table 1–1:**

(1) The default `<Nios II EDS installation directory>` location is `C:\altera\<version>\nios2eds`
8. Click **Close**. You return to the Quartus II Programmer window.

9. Click **Auto Detect**. The device on your board is detected automatically.

10. Click the first entry to highlight it. Refer to Figure 1–3 for the location of the first entry.

11. Click **Change File**.

12. Browse to `<your project directory>` and select the `.sof` programming file for your design shown in Table 1–1.

13. Click **OK**.

14. Turn on **Program/Configure** the programming file, as shown in Figure 1–3.
15. Click **Start**.

The **Progress** meter sweeps to 100% as the Quartus II software configures the FPGA. When configuration is complete, the FPGA is configured with the Nios II system, but it does not yet have a C program in memory to execute.

**Nios II SBT for Eclipse Build Flow**

The Nios II SBT for Eclipse is an easy-to-use GUI that automates build and makefile management, and integrates a text editor, debugger, the Nios II flash programmer, and the Quartus II Programmer. Software application templates included in the GUI make it easy for new software programmers to get started quickly.

In this section, you use the Nios II SBT for Eclipse to compile a simple C language example software program to run on the Nios II standard system configured in the FPGA on your development board. You create a new software project, build it, and run it on the target hardware. You also edit the project, re-build it, and set up a debugging session.

For a complete tutorial on using the Nios II SBT for Eclipse to develop programs, refer to the software development tutorial in the *Getting Started with the Graphical User Interface* chapter of the **Nios II Software Developer’s Handbook**.

**Create the Hello World Example Project**

In this section, you create a new Nios II application project from an installed example. To begin, perform the following steps in the Nios II SBT for Eclipse:
1. Return to the Nios II SBT for Eclipse.

   You can close the Quartus II Programmer or leave it open in the background if you want to reload the processor system onto your development board quickly.

2. Create a new workspace in the `<your project directory>` so that the software resides under its hardware project. To do so, perform the following steps:
   a. On the File menu, point to Switch Workspace, and click Other. The Workspace Launcher dialog box appears.
   b. Click Browse to display the Select Workspace Directory dialog box, then navigate to `<your project directory>`.
   c. Click Make New Folder.
   d. Type `eclipse_workspace` and press Enter.
   e. Click OK to exit the Select Workspace Directory dialog box.
   f. Click OK to exit the Workspace Launcher dialog box. The Nios II Software Build Tools for Eclipse restarts in the new workspace.

3. On the File menu, point to New, and click Nios II Application and BSP from Template. The Nios II Application and BSP from Template wizard appears.

4. For SOPC Information File name, browse to `<your project directory>` and open the SOPC Information File (.sopcinfo) for your design shown in Table 1–1.

   Every Nios II software project needs a system description of the corresponding Nios II hardware system. For the Nios II SBT for Eclipse, this system description is contained in a .sopcinfo file.

5. In the Project name box, type `my_first_nios_software_project`.

6. In the Templates list, select the Hello World project template. Figure 1–4 shows the wizard.
7. Click Finish. The Nios II SBT for Eclipse creates the my_first_nios_software_project project and returns to the Nios II perspective.

8. In the Project Explorer view, expand my_first_nios_software_project. Double-click hello_world.c to view the source code. Figure 1–5 shows the Nios II perspective.
When you create a new project, the Nios II SBT for Eclipse creates the following new projects in the Project Explorer view:

- **my_first_nios_software_project** is the application project. This project contains the source and header files for your application.
my_first_nios_software_project_bsp is the board support package (BSP) for your Nios II system hardware. The BSP includes the following details:

- Component device drivers for your Nios II hardware system
- newlib C library, which is a richly-featured C library for embedded systems
- Nios II software packages
  - Nios II hardware abstraction layer (HAL)
  - NicheStack TCP/IP Network Stack, Nios II Edition
  - Nios II host file system
  - Nios II read-only zip file system
  - Micrium’s MicroC/OS-II real-time operating system (RTOS)
- system.h, which is a header file that encapsulates your hardware system
- alt_sys_init.c, which is an initialization file that initializes the devices in the system
- linker.h, which is a header file that contains information about the linker memory layout.

**Build and Run the Program**

Perform the following steps to build and run the program:

1. To build the program, right-click the my_first_nios_software_project project in the Project Explorer view, and click Build Project. The Build Project dialog box appears and the Nios II SBT for Eclipse begins compiling the project. When compilation completes, the message “Build completed” appears in the Console view. The completion time varies depending on your system.

2. To run the program, right-click my_first_nios_software_project, point to Run As, and click Nios II Hardware. The Nios II SBT for Eclipse downloads the program to the FPGA on the target board and executes the code. The message “Hello from Nios II!” displays in the Nios II Console view.

   If the Run Configurations dialog box appears, click the Target Connection tab and verify the connection to the board. If no connection is shown, click Refresh Connections. When a connection appears, click Run. For more information about run and debug configurations, refer to the Getting Started with the Graphical User Interface chapter of the Nios II Software Developer’s Handbook.

Figure 1–6 shows the Nios II Console view at the right side of the Nios II perspective.
Now that you have created, compiled, and run your first software program, you can perform additional operations, such as configuring the BSP properties, editing and rebuilding the application, and debugging the source code.

**Edit and Re-Run the Program**

You can modify the source code in the Nios II SBT for Eclipse, rebuild the project, run the program, and observe your changes executing on the target board. In this section, you add code that makes an LED on your development board blink.

For more information regarding how the LED blinks, refer to “Debugging the Application” on page 1–11.

Perform the following steps to modify and re-run the program:

1. In the `hello_world.c` file, replace the existing code with the following code:

```c
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
int main()
{
    int count = 0;
    int delay;
    printf("Hello from Nios II!
");
```
while(1)
{
    IOWR_ALTERA_AVALONPIO_DATA(LEDPIO_BASE, count & 0x01);
    delay = 0;
    while(delay < 2000000)
    {
        delay++;
    }
    count++;
    return 0;
}

2. Save the file.

3. Right-click my_first_nios_software_project in the Project Explorer view, point to Run As, and click Nios II Hardware.

    You do not need to build the project manually; the Nios II SBT for Eclipse automatically rebuilds the program before downloading it to the FPGA.

4. Observe the LED blinking on your development board.

Debugging the Application

To debug your application, perform the following steps:

1. Right-click my_first_nios_software_project in the Project Explorer view, point to Debug As, and click Nios II Hardware.

    If the Debug Configurations dialog box appears, click the Target Connection tab and verify the connection to the board. If no connection is shown, click Refresh Connections. When a connection appears, click Debug. For more information about run and debug configurations, refer to the Getting Started with the Graphical User Interface chapter of the Nios II Software Developer’s Handbook.

2. If the Confirm Perspective Switch message box appears, click Yes. The context switches to the Nios II Debug perspective.

    After a moment, the main() function appears in the editor. A blue arrow next to the first line of code indicates that execution stopped at that line.

3. On the Run menu, click Resume to resume execution.

    When debugging a project in the Nios II SBT for Eclipse, you can pause, stop, or single step the program, set breakpoints, examine variables, and perform many other common debugging tasks.

    To return to the Nios II perspective from the Nios II Debug perspective, click the two arrows (>>) in the upper right corner of the GUI, and click Nios II.

    For more information about debugging software projects in the Nios II SBT for Eclipse, refer to the Getting Started with the Graphical User Interface chapter of the Nios II Software Developer’s Handbook.
Why the LED Blinks

The Nios II system description header file, system.h, contains the software definitions, name, locations, base addresses, and settings for all of the components in the Nios II hardware system. The system.h file resides in the my_first_nios_software_project_bsp directory. Open the system.h file and locate the LEDPIO_BASE macro.

LEDPIO_BASE contains the base address of the PIO peripheral controlling the LEDs. The Nios II processor controls the PIO ports (and therefore the LED) by reading and writing to the register map. For the PIO, there are four registers: data, direction, interruptmask, and edgecapture. To turn the LED on and off, the application writes to the PIO data register.

The PIO core has an associated software file altera_avalon_pio_regs.h. This file defines the core’s register map, providing symbolic constants to access the low-level hardware. The altera_avalon_pio_regs.h file resides in the my_first_nios_software_project_bsp directory in the drivers\inc subdirectory.

When you include the altera_avalon_pio_regs.h file, several useful functions that manipulate the PIO core registers are available to your program. In particular, the function IOWR_ALTERA_AVALON_PIO_DATA(base, data) can write to the PIO data register, turning the LED on and off.

The PIO is just one of many SOPC Builder peripherals that you can use in a system. To learn about the PIO core and other embedded peripheral cores, refer to Volume 5: Embedded Peripherals of the Quartus II Handbook.

When developing your own designs, you can use the software functions and resources that are provided with the Nios II HAL.

Refer to the Nios II Software Developer’s Handbook for extensive documentation on developing your own Nios II processor-based software applications.

Board Support Package

This section explores configuring your BSP.

For BSP properties changes to take affect, you must regenerate your BSP before re-running your program by either clicking Generate in the BSP Editor or by right-clicking the my_first_nios_software_project_bsp project in the Project Explorer view, pointing to Nios II, and clicking Generate BSP.

To access the most common BSP properties, perform the following steps:

1. In the Nios II SBT for Eclipse, right-click my_first_nios_software_project_bsp and click Properties. The Properties for my_first_nios_software_project_bsp dialog box appears.

2. Click Nios II BSP Properties. The most-common settings related to how the program interacts with the underlying hardware appear. Figure 1–7 shows the Nios II BSP Properties dialog box.
An extensive set of build properties and options are available using the BSP Editor. To access all the BSP properties, perform the following steps:

1. In the lower right corner of the Properties for my_first_nios_software_project_bsp dialog box, click BSP Editor. The Nios II BSP Editor appears. Figure 1–8 shows the Nios II BSP Editor.

You can also access the Nios II BSP Editor from the Nios II menu.
2. Click the **Settings** tab to see the available settings. For example, you can set the interfaces to use for `stdio`, `stdin`, and `stderr`.

3. Click the **Software Packages** tab to see the software packages, such as files systems, graphics libraries, network stacks, available in your BSP.

4. Click the **Drivers** tab to see the available drivers for the Altera-provided intellectual property (IP) in your system.

5. Click the **Linker Script** tab to see the linker section memory assignments. The linker section assignments determine what memory is used to store the compiled executable program when the `my_first_nios_software_project` program runs. Figure 1–9 shows the **Linker Script** tab of the Nios II BSP Editor.
6. Click the **Target Generation File** tab to see the files that get added to your BSP at build time.

7. If you have made any changes to your BSP, click **Generate** to update your BSP.

8. On the File menu, click **Exit**.

9. Click **OK** to close the **Properties for my_first_nios_software_project_bsp** dialog box.

**Next Steps**

The following documents provide next steps to further your understanding of the Nios II processor:

- **Developing Software for Nios II Processors**—These short, online software tutorials walk you through the basics of developing software for the Nios II processor. Access these tutorials on the **Embedded Training Resources** page of the Altera website.
Nios II Software Developer’s Handbook — This handbook provides a complete reference on developing software for the Nios II processor.

The “Getting Started” section of the Getting Started with the Graphical User Interface chapter of the Nios II Software Developer’s Handbook — This tutorial teaches in detail how to use the Nios II SBT for Eclipse to develop, run, and debug new Nios II application projects.

Nios II Processor Reference Handbook — This handbook provides a complete reference for the Nios II processor hardware.

Volume 4: SOPC Builder of the Quartus II Handbook — This volume provides a complete reference on using SOPC Builder, including building memory subsystems and creating custom components.

Volume 5: Embedded Peripherals of the Quartus II Handbook — This volume contains details on the peripherals provided with the Nios II Embedded Design Suite.

For a complete list of documents available for the Nios II processor, refer to the Literature: Nios II Processor page of the Altera website.
**Document Revision History**

The following table shows the revision history for this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 2010</td>
<td>2.0</td>
<td>Revised for Nios II Software Build Tools for Eclipse.</td>
</tr>
<tr>
<td>July 2008</td>
<td>1.4</td>
<td>Removed command line flow to simplify tutorial.</td>
</tr>
<tr>
<td>May 2007</td>
<td>1.3</td>
<td>Minor fixes.</td>
</tr>
<tr>
<td>May 2007</td>
<td>1.2</td>
<td>Minor fixes.</td>
</tr>
<tr>
<td>May 2007</td>
<td>1.1</td>
<td>Minor fixes.</td>
</tr>
<tr>
<td>May 2007</td>
<td>1.0</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>

**How to Contact Altera**

For the most up-to-date information about Altera products, refer to the following table.

<table>
<thead>
<tr>
<th>Contact &lt;Italic&gt; (Note 1)</th>
<th>Contact Method</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical support</td>
<td>Website</td>
<td><a href="http://www.altera.com/support">www.altera.com/support</a></td>
</tr>
<tr>
<td>Technical training</td>
<td>Website</td>
<td><a href="http://www.altera.com/training">www.altera.com/training</a></td>
</tr>
<tr>
<td></td>
<td>Email</td>
<td><a href="mailto:custrain@altera.com">custrain@altera.com</a></td>
</tr>
<tr>
<td>Product literature</td>
<td>Website</td>
<td><a href="http://www.altera.com/literature">www.altera.com/literature</a></td>
</tr>
<tr>
<td>Non-technical support (General)</td>
<td>Email</td>
<td><a href="mailto:nacomp@altera.com">nacomp@altera.com</a></td>
</tr>
<tr>
<td>(Software Licensing)</td>
<td>Email</td>
<td><a href="mailto:authorization@altera.com">authorization@altera.com</a></td>
</tr>
</tbody>
</table>

**Note to Table:**
(1) You can also contact your local Altera sales office or sales representative.

**Typographic Conventions**

The following table shows the typographic conventions this document uses.

<table>
<thead>
<tr>
<th>Visual Cue</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold Type with Initial Capital Letters</strong></td>
<td>Indicates command names, dialog box titles, dialog box options, and other GUI labels. For example, <strong>Save As</strong> dialog box. For GUI elements, capitalization matches the GUI.</td>
</tr>
<tr>
<td><strong>bold type</strong></td>
<td>Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, <code>\qdesigns</code> directory, <code>d:</code> drive, and <code>chiptrip.gdf</code> file.</td>
</tr>
<tr>
<td><strong>Italic Type with Initial Capital Letters</strong></td>
<td>Indicates document titles. For example, <strong>AN 519: Stratix IV Design Guidelines</strong>.</td>
</tr>
</tbody>
</table>
## Typographic Conventions

<table>
<thead>
<tr>
<th>Visual Cue</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Italic type</em></td>
<td>Indicates variables. For example, ( n + 1 ). Variable names are enclosed in angle brackets ((&lt;&gt;)). For example, (&lt;file name&gt;) and (&lt;project name&gt;.pof) file.</td>
</tr>
<tr>
<td>Initial Capital Letters</td>
<td>Indicates keyboard keys and menu names. For example, the Delete key and the Options menu.</td>
</tr>
<tr>
<td>“Subheading Title”</td>
<td>Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, “Typographic Conventions.”</td>
</tr>
<tr>
<td><strong>Courier type</strong></td>
<td>Indicates signal, port, register, bit, block, and primitive names. For example, ( data1, tdi, ) and ( input ). The suffix ( n ) denotes an active-low signal. For example, ( resetn ). Indicates command line commands and anything that must be typed exactly as it appears. For example, ( c:\qdesigns\tutorial\chiptrip.gdf ). Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword ( SUBDESIGN )), and logic function names (for example, ( TRI )).</td>
</tr>
<tr>
<td>1., 2., 3., and a., b., c., and so on</td>
<td>Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.</td>
</tr>
<tr>
<td>■ ■ ■</td>
<td>Bullets indicate a list of items when the sequence of the items is not important.</td>
</tr>
<tr>
<td>️</td>
<td>The hand points to information that requires special attention.</td>
</tr>
<tr>
<td>🚨</td>
<td>A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.</td>
</tr>
<tr>
<td>🚨</td>
<td>A warning calls attention to a condition or possible situation that can cause you injury.</td>
</tr>
<tr>
<td>➔</td>
<td>The angled arrow instructs you to press the Enter key.</td>
</tr>
<tr>
<td>🚹</td>
<td>The feet direct you to more information about a particular topic.</td>
</tr>
</tbody>
</table>