Overview

What does it take to develop a System-Level Design?

This lab teaches you how to create a system implemented in programmable logic. You build a processor-based hardware system and run software on it. As the lab progresses, you will see how quick and easy it is to build entire systems using Altera’s SOPC Builder to configure and integrate pre-verified IP blocks.

Note: This lab guide requires a DE0 Nano Evaluation Board.

Lab Notes:

Many of the names that the lab asks you to choose for files, components, and other objects in this exercise must be spelled exactly as directed.

This nomenclature is necessary because the pre-written software application includes variables that use the names of the hardware peripherals. Naming the components differently can cause the software application to fail.

There are also other similar dependencies within the project that require you to enter the correct names.
MODULE 1: Getting Started

Module Objective

Your first objective is to ensure that you have all of the items needed and to install the tools so that you are ready to create and run your design.

List of required items:
⇒ DE0 Nano Evaluation Board with USB cable
⇒ Design Software (Quartus II design software v10.1, Nios II EDS 10.1)
⇒ Intel Pentium III or compatible Windows PC, running at 866MHz or faster, with a minimum of 512MB of RAM
⇒ Lab Design Files

1.1 Acquire the DE0 Nano Development Board

This development kit can be ordered from http://www.terasic.com
1.2 Install the Design Software

You will need to install **BOTH** of the following design software packages:

1) **Quartus II design software v10.1** – FPGA synthesis and compilation tool that contains SOPC Builder and the MegaCore IP library with the Nios II processor IP core
2) **Nios II EDS v10.1** – A complete integrated development environment for software development

The design software is available via the Altera Complete Design Suite DVD or by downloading from the web.

*If installing from the DVD-ROM, please skip ahead to the DVD installation instructions below.*

*If you already have both Quartus II and the Nios II EDS installed on your machine, you may skip ahead to Section 1.3 to install the DE0 Nano files and drivers.*

**INSTALLING FROM THE WEB:**

The Web Edition can be downloaded from
https://www.altera.com/download/software/quartus-ii-we

Remember to download and install **BOTH** of the design software packages.

- The Subscription package requires a paid subscription license, which will entitle you to a full license for all of Altera’s IP Base Suite.
- For this lab, either the Subscription Edition or Web Edition is acceptable.
1.3 Extract the DE0 Nano Installation and Lab Files

Extract the DE0Nano_sopc_builder_lab.zip and simple_pwm.zip files into a folder on your PC. Make sure that there are **NO SPACES** in the directory path.

1.4 Install the DE0 Nano USB Device Drivers

After the Quartus II and Nios II software packages are installed, you can plug the DE0 Nano board into your USB port. *(NOTE: If you are using Windows 7 (32-bit), please refer to Appendix A for instructions on disabling Digital Driver Signing checks.)*

Your Windows PC will find the new hardware and then the “Found New Hardware Wizard” will come up.

Select “Install from a list or specific location (Advanced)” and continue through the wizard.

In the next dialogue box point the wizard to the drivers which can be found in your DE0 Nano installation directory under “%<install directory>%\driver\”.

If Windows presents you with a message that the drivers have not passed Windows Logo testing, please click “Continue Anyway”.

CONGRATULATIONS!!

You have just completed all the setup and installation requirements and are now ready to start your first system-level design.
 MODULE 2: Design the System

Module Objective

Architecting a system involves specifying the design requirements and developing a suitable design strategy to address each requirement. Typically, design requirements begin with customer requirements and become inputs to system definition. System definition is hence the first step in the design flow process. For this lab, our objective is to define a basic embedded processor-system for the DE0 Nano kit.

2.1 Design Flow

The above diagram depicts the typical flow for system design. System definition is performed using SOPC Builder.

The results are two-fold:

- System description that the Nios II Integrated Development Environment, the software design tool, uses to create a new project for the software application.
- HDL files for the system that are used by the Quartus II FPGA design software to compile and generate the hardware system.
Design the System

The output of the Hardware Flow is an FPGA image that is used to configure the FPGA. The output of the Software Flow is an executable from which the Nios II processor executes instructions.

The following steps summarize the design flow that you will follow in this lab:

1. Launch the Quartus II software
   a. Set up a new Quartus II project
   b. Assign the device
   c. Assign preliminary timing constraints
   d. Make pin assignments
2. Launch SOPC Builder and build an SOPC system
   a. Select and configure IP cores
   b. Make connections, assign clocks
   c. Set arbitration priorities
3. Generate the system to create
   a. HDL for the entire SOPC system
   b. A system description file that software development tools use to build the board support package
4. Complete the Quartus II Project
   a. Generate a schematic for the SOPC system
   b. Add the SOPC system to the top level
   c. Complete the pin assignments
   d. Compile to generate a FPGA programming file (*.sof)
5. Launch Nios II EDS and develop the software application
   a. Add source files
   b. Configure build properties
   c. Build application to generate the executable (*.elf)
6. Use the Quartus II Programmer to download the FPGA image (*.sof)
7. Use the Nios II Software Build Tools for Eclipse to download and run the executable (*.elf) for the software application
2.2 Design Requirements

Examine the components on the DE0 Nano board hardware:

The board has been designed for very low cost and the design has been kept very simple. As such there are very few peripherals available, but what we do have is 8 LEDs and an external SDRAM.

<table>
<thead>
<tr>
<th>Design Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>A user shall be able to use a combination of keys and switches to:</td>
</tr>
<tr>
<td>- Control LEDs</td>
</tr>
<tr>
<td>- Communicate via JTAG UART</td>
</tr>
<tr>
<td>- Control parameters such as</td>
</tr>
<tr>
<td>- Step count</td>
</tr>
<tr>
<td>- Right or left increment</td>
</tr>
<tr>
<td>- Set step up or step down delay</td>
</tr>
</tbody>
</table>
2.3 Design Strategy

SOPC Builder provides a menu of standard hardware components that you add to the system. If the required component is not present in the SOPC Builder menu, you may purchase one from a third party or build one and add it to the system as a custom component.

<table>
<thead>
<tr>
<th>Design Requirement</th>
<th>Design Strategy</th>
<th>Standard or Custom?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control LEDs</td>
<td>PIO component</td>
<td>Standard</td>
</tr>
<tr>
<td>Control the intensity of an LED using pulse width modulation</td>
<td>Will need a PWM core which can read an intensity custom value from a register and output a resulting modulated waveform</td>
<td>Custom</td>
</tr>
<tr>
<td>Respond to stimulus and count in binary</td>
<td>A processor</td>
<td>Standard</td>
</tr>
<tr>
<td>Keep track of hardware and software builds</td>
<td>System ID Peripheral</td>
<td>Standard</td>
</tr>
<tr>
<td>Generate clocks for my design blocks and external chips</td>
<td>PLL</td>
<td>Standard</td>
</tr>
<tr>
<td>Some memory to hold program and data</td>
<td>Use the on-board SDRAM memory for this, will need an SDRAM Controller</td>
<td>Standard</td>
</tr>
<tr>
<td>Communicate with the processor</td>
<td>JTAG UART communication interface</td>
<td>Standard</td>
</tr>
<tr>
<td>Generate interrupts and measure system performance</td>
<td>Interval Timers</td>
<td>Standard</td>
</tr>
</tbody>
</table>

2.4 System Schematic

Now you have a design strategy to address each of the design requirements. The system will consist of a combination of custom and standard components. Below is a sketch of this concept:

As systems get larger and more complicated, it is easier to design at a higher level of abstraction using (and reusing) IP blocks. SOPC Builder automatically generates the system interconnect, i.e. the glue logic required to connect the design blocks together.

The system interconnect manages dynamic bus-width matching, interrupt priorities, arbitration etc.

CONGRATULATIONS!!

By understanding the design flow, design requirements and design strategy you have developed your system architecture.
Set up the Quartus II Project

Module Objective

In this module you create a new Quartus II project to contain the SOPC Builder system. The top level is a schematic file, which at this stage is a placeholder containing some minimal reset logic, i.e. a counter that issues a reset to the SOPC system in response to a hard reset.

In the last module several custom components were identified for the system. These components are provided as part of the design files for this lab. In this module you include paths to these components and also specify I/O constraints and settings for this design by executing a Tcl script.

3.1 Create New Quartus II Project

- Launch the Quartus II software from Start -> All Programs -> Altera.
- Click on File -> New Project Wizard. This will launch the New Project Wizard.
- For the working directory for the project, click the Browse button indicated by the “…” symbol and navigate to the folder “DE0Nano_sopc_builder_lab” located in the unzipped lab design files. This will be the working directory for your project.
- Name the project “DE0Nano_lab”.
- For the top-level entity click on “…” DE0Nano_lab_top
- Click Next
3.2 Add Files to the Project

- In the Wizard window page 2 of 5 you will add files to the new project.
- Click the **Browse** button and navigate to the project directory and **open** the folder entitled “DE0_Nano_sopc_builder_lab”.

- Select and click **Open** to add the appropriate top-level file
  - DE0Nano_lab_top.v: This is the top-level Verilog entity for the Quartus II Project.
  - Click **Add** to add the files listing chart.
3.3 Specify Family and Device Settings

In this page you select Cyclone IV E **EP4CE22F17C6** device, which is the device mounted on the DE0 Nano circuit board. You can use the “Show in ‘Available device’ list” option to filter the list of available devices to make selection easier.

Click **Next**.

3.4 Select EDA Tool Settings

- Select <None> for all of the options. Click **Next**.
- You will see a **Summary** page. Click **Finish**.
3.5 Import CSV Script

The I/O pin constraints have been programmed into a csv file in order to set up the Quartus II project properly.

Under the Assignments menu, select “Import Assignments”. Then pick the DE0_Nano_lab.csv from the project directory and press OK.

Under the Assignments menu choose “Pin Planner”.

In your Quartus II lower console you will see a message “Info: Import completed. 96 assignments were written (out of 96 read). 0 non-global assignments were skipped because of entity name mismatch.”.

CONGRATULATIONS!!

Your Quartus II project is set up. You are ready to start building your SOPC system.
Module Objective

In this module you add the standard and custom components to the system, make connections where required, assign the clocks, set arbitration priorities and generate the system.

4.1 Launch SOPC Builder

- From the Tools menu, select “SOPC Builder”. There may be a slight delay while the SOPC Builder application launches.
- In the “Create New System” dialog, select Verilog and enter the system name as “DE0_Nano_system”.
- Click OK.

Perform the following instructions to build the system.

It is helpful to have the rough sketch of your system handy so you can follow along.

4.3 Build the SOPC System

1. Set the external clock source to 50.0 MHz

   Reason: The clock coming into the FPGA is sourced by an on-board 50MHz crystal oscillator. This clock source will feed our SOPC Builder system.

   Click into the clk_0 field and rename it from clk_0 to ext_clk.

1. Add a System ID

   Reason: This is a VERY IMPORTANT peripheral to have in your system. It allows the Nios II development tools to validate that the software application is being built for the correct hardware system.

   - From the System Contents menu, select Peripherals -> Debug and Performance -> System ID Peripheral.
   - Double click to add the component to the system.
   - The sysid dialog box appears. Click Finish.

   A component entitled “sysid_0” should appear under Module Name. Rename as “sysid”. (You can right click to bring up a menu with a rename option.)

   The component must be named “sysid” to be compatible with Nios II software drivers and build tools.
An error will appear in the bottom console indicating that the sysid peripheral is not connected to a master. Ignore this for now. We will address connections at a later stage.

2. Add an Avalon ALTPLL

**Reason:** This peripheral instantiates a PLL which will generate the clocks for the system.

From the **System Contents** menu, expand **PLL** and double click on **Avalon ALTPLL**.

“General/Modes” tab (Page 1) of PLL MegaWizard. Accept defaults for this page. The clock input to the PLL is 50 MHz.

This source is provided by the oscillator on the DE0 Nano board.

“Inputs/Lock” tab (Page 2) : Confirm that both “Create an ‘areset’ input to asynchronously reset the PLL” and “Create ‘locked’ output” options are unchecked.

Accept all other defaults.

Pages 3-5 : Accept all defaults.
Build the SOPC System

“Output Clocks: clk c0” (Page 6) : Click “ Enter output clock frequency “. Configure c0 as 50 MHz output and a -60 deg clock phase shift. This clock will be used as the SDRAM clock, clocking the SDRAM.

“Output Clocks: clk c1” (Page 6) : Click “ Enter output clock frequency “. Configure c0 as 50 MHz output. This clock will be used as the main system clock, clocking almost the entire system logic.
• Click **Finish**. This will take you to the summary tab.
• Click **Finish** again to close the ALTPLL MegaWizard.

⇒ Rename the Avalon ALTPLL component from "altpll_0" to "pll"
⇒ PLL Cleanup and frequency settings: Ensure that the name of the PLL is "pll".
⇒ In the Clock Settings window (top right), the pll clock and the source clock should appear. Rename the clocks as follows:
   1. pll.c0 = sdram_clk
   2. pll.c1 = sys_clk

3. **Configure clocks for the initial components**

   At this point there are 2 components added to the system.
From the drop-down list in the **Clock** column, ensure that the **PLL** is set up with the **ext clk** source, and the **sysid** is driven by the **sys clk** source.

Since the **sysid** is driven by ext_clk, click on that field and change the selection to **sys_clk** as shown below:

4. Add a Nios II CPU

**Reason**: The demo will be controlled by a software application written in C. A CPU will run this software application in the FPGA.

- From the **Component Library** pane, under the Library, Expand **Processors** and double click on **Nios II Processor**.
- Ensure that the Nios II/f core is selected. All the defaults should be accepted.
- Click **Finish**.
- Rename the component “cpu”.

Occasionally save your work using **Save** on the **File** menu.
5. Add SDRAM Controller

**Reason**: The DE0 Nano board has an SDRAM device that can be used as additional memory for code or data storage.

This peripheral defines the timing controller for driving the read and write transactions to the external SDRAM.

- From the **System Contents** menu, expand **Memories**, expand **SDRAM** and double click on **SDRAM Controller**.
- Now we need to configure the SDRAM attributes for the communication with the chip.

- Click on finish and rename it to **s dram**.
6. Add JTAG UART Peripheral

**Reason**: Many software developers like to have access to a debug serial port from the target to leverage printf debugging, input control commands, log status information, etc. The JTAG UART peripheral connects to the debugger console and is useful for these purposes.

- From the **System Contents** menu, expand **Interface Protocols**, expand **Serial** and double click on **JTAG UART**.
- The default settings are acceptable. Click **Finish**.
- Rename as “**jtag_uart**”.

![JTAG UART peripheral settings](image_url)
7. Add Interval Timer Peripheral

**Reason**: Many software applications require periodic interrupts to maintain various time bases and timing requirements within the application.

- From the **System Contents** menu, expand **Peripherals**, expand **Microcontroller**
- Expand **Peripherals** and double click on **Interval Timer**.
- Accept the default settings. Click **Finish**.
- Rename the component “sys_timer”. Ensure that sys_timer/s1 is connected to the cpu/data_master.
8. Add a second Interval Timer Peripheral

Reason: The Nios II HAL library provides a high resolution timer facility that allows software applications to measure time at the system clock rate. A second timer peripheral is useful for this.

- From the System Contents menu, expand Peripherals, expand Microcontroller Peripherals
- Double click on Interval Timer.
- The resolution should be changed to 1 us. Click Finish.
- Rename the component “hires_timer”. The hires_timer/s1 should already be connected to the cpu/data_master.

9. Add PIO Peripheral for LEDs

Reason: The DE0 Nano board has 8 LEDs on it. You can drive these LEDs with an output PIO peripheral. We will drive 7 of the LEDs with a PIO peripheral. (The 8th LED will be controlled by a custom PWM peripheral added in the next step.)

- From the System Contents menu, expand Peripherals, expand Microcontroller Peripherals
- Double click on PIO (Parallel I/O).
- Set the “Width” to 7 bits. Set “Direction” to “Output ports only”. Click Finish.
- Rename the peripheral “LED_G”.
10. Add PWM Peripheral

**Reason**: We will use the custom PWM component to control the intensity of the 8 LEDs.

- From the *Project* section of the *Components Library*, expand *DE0 Nano Components* and double click on *Simple PWM*.
- Rename the peripheral “*LED_pwm*”.

At this point all the components to the SOPC system have been added. Now you need to resolve the lingering system validation errors that have arisen during the design.

### 4.4 System Configuration

1. **Set Base Addresses and Interrupt Priorities**

SOPC Builder provides two easy menus that help clean up address map issues and interrupt priority issues.

From the *System* menu, choose *Auto-Assign Base Addresses*. The tool will assign appropriate base addresses for the components by taking their widths into consideration.

From the *System* menu, choose *Auto-Assign IRQs*. The tool will update the IRQ mapping accordingly.

At this point you should only be left with a couple of information messages and reminders that you have yet to specify the CPU reset and exception address configuration.

2. **Clock source for each component**

Ensure that only the PLL has *ext_clk* selected in the Clock column, and all other components have *sys_clk* selected for their clock sources.
3. Nios II Boot Configuration

In the event of a reset, the software must begin executing from a predefined memory location. This is set by setting the reset vector.

Similarly when a software exception event occurs the software must jump to a pre-defined location where the exception handling software resides. This location is set by setting the exception vector.

- Double click on the **cpu** peripheral to launch the “Nios II Processor Parameter Settings” GUI.
- Set the **Reset Vector** to point to the **sdram** memory with an offset of 0x0.
- When the Nios II processor comes out of reset, it will begin executing software at this memory location.
- Set the **Exception Vector** to point to the **sdram** memory with an offset of 0x20.
- When the Nios II processor experiences software exceptions or interrupts, it will jump to this location in memory.

![Vector Settings](image)

This concludes the system configuration. At this point you should have addressed all the system validation issues and can now generate the SOPC Builder System.
4.5 Generate the System

Please Double-check to make sure that all the component names in your SOPC system match the component names shown above. Do not worry if the base and end addresses do not match exactly.

Click the Generate button. SOPC Builder will now create:

- The HDL for the various components in your system
- System interconnect to connect the components together
- System description file used by the software development tools (the Nios II SBT) to build the software project

Once your system has been successfully generated you will see the info message “System generation was successful”.

Exit SOPC Builder by clicking the Exit button.

CONGRATULATIONS!! You have just built your first SOPC system!
Module Objective

In this module you complete the Quartus II project by adding the generated SOPC system to the top-level entity. Compile in the Quartus II software to perform analysis, synthesis, fitting, place and route as well as timing analysis. At the end of the compilation, an FPGA image or SDRAM object file (*.SOF) will be generated. The FPGA image can be downloaded to the DE0 Nano board, at which point the on-board FPGA will function as a processor custom-made for your application.

5.1 Complete the Quartus II Project

In the Project Navigator window pane in the Quartus II software, select the Files tab and open the DE0 Nano_lab_top.v by double clicking it.

The pins required by the design as well as some reset logic are present, but the main block of logic representing the SOPC Builder system is missing. Add the SOPC Builder system instance to this top level.

- From the Edit Menu, click on Insert Template.
- In the Language Templates pane expand the MegaFunctions folder and then expand the Instances, choose the DE0_Nano_system_inst.v
- Then click OK.
Map the ports in the top file so it looks like this:

```vhdl

vuln reset_n;
assign reset_n = '1'b1;

//Example instantiation for system 'DE0 Nano_system'
DE0_Nano_system DE0_Nano_system_inst (
  .ext_clk (CLOCK_60),
  .out_port_from_the_LED_S (LED[6:0]),
  .pwm_out_from_the_led_PWM (LED(7)),
  .reset_n (reset_n),
  .sdram_clk (sdram_clk),
  //in Port to the Button
  .rs_addr_from_the_sdram(DRAM_ADDR),
  .rs_be_from_the_sdram(DRAM_BA),
  .rs_cas_n_from_the_sdram(DRAM_CAS_N),
  .rs_cdo_from_the_sdram(DRAM_CD),
  .rs_dq_to_end_from_the_sdram(DRAM_DQ),
  .rs_dqm_from_the_sdram(DRAM_DQM),
  .rs_we_n_from_the_sdram(DRAM_WE_N)
);

assign DRAM_CLK = sdram_clk;
```

- Click the **Start Compilation** button on the Quartus II tool bar.

The Quartus II software will take a few minutes to compile the design. There should be no errors in the compile, and you should see the successful completion dialog when it is finished. You will see some warnings that relate to the files from the automatically generated system, missing assignments/features and incomplete pin assignments but these will not affect the functionality of the system.

The output of the compilation is a SOF file entitled “DE0 Nano_lab.sof” if you have a Nios II license or “DE0 Nano_lab_time_limited.sof” if you do not have a license.

### 5.2 Download the FPGA configuration

Ensure that your DE0 Nano kit is plugged into your PC USB port and launch the Quartus II Programmer to configure the FPGA.

- Click on the **Programmer** icon on the Quartus II desktop, or alternatively open the **Programmer** from the **Tools** menu.

  ⇒ In Quartus II Programmer click **Hardware Setup**. In the **Currently selected hardware** drop box select **USB-Blaster**.
  ⇒ Click **Close**.
  ⇒ Click on **Auto Detect**.
  ⇒ After clicking the **Auto Detect** button you should find that the programmer detects the EPCS device on your DE0 Nano board.
  ⇒ Double click on the `<none>` in the **File** field, or select `<none>` and click on the **Change File** button.
  ⇒ Select DE0Nano_lab_***.sof. Click **Open**.
Complete the Quartus II Project

- After selecting your SOF file, click on the **Program/Configure** checkbox.
- Press the **Start** button to program the FPGA.

After programming the FPGA the progress indicator should indicate 100% complete, and there should be no error messages displayed.

**NOTE:** If you do not have a license for the Nios II processor then your system would have generated the Nios II in OpenCore Plus evaluation mode and your sof programming file will be time-limited. If you are running with a time-limited SOF file, then this window pops up on the Quartus II Programmer.

Just leave this up and **do not press “Cancel”** until you are finished using the hardware design that you just downloaded. Closing this dialog will halt the Nios II CPU inside the FPGA.

**CONGRATULATIONS!!**

You have just compiled and downloaded the FPGA image onto the target. The processor is ready to run, so all you need to do now is develop the software application and download it to the target.
MODULE 6: Build the Software Application

Module Objective

In this module you use the Nios II Software Build Tools (SBT) for Eclipse to develop the software application that will run on your system. You will create a new software application project, add the software source files to the project, configure the project and build it. The result of the build is an executable (ELF). The application will be downloaded into memory from where it will be executed.

6.1 Launch the Nios II Software Build Tools for Eclipse

Launch the Nios II SBT from the Start -> All Programs -> Altera -> Nios II EDS -> Nios II Software Build Tools for Eclipse or you can launch it from SOPC Builder-> Nios II menu.

1. Initialize Eclipse workspace

When Eclipse first launches, a dialogue box appears asking what directory it should use for its workspace. It is useful to have a separate Eclipse workspace associated with each hardware project that is created in SOPC Builder.

Browse to the directory that you created the Quartus II project in and click Make New Folder to create a folder for your software project. Name the folder “eclipse_workspace”.

After selecting the workspace directory, click “OK” and Eclipse will launch and the workbench will appear in the Nios II perspective.
6.2 Create a new software project in the SBT
Select File -> New -> Nios II Application and BSP from Template.

- To set the SOPC Information File, click the Browse button to locate the DE0 Nano_system.sopcinfo file located in the Quartus II project directory.
- Set the name of the Application project to "DE0_Nano_led_control".
- Select the Blank Project template under Project Template.
- Click the Finish button.

The tool will create two new software project directories
Each Nios II application has 2 project directories in the Eclipse workspace.

a. The application software project itself - this where the application lives.

b. The second is the Board Support Package (BSP) project associated with the main application software project.

This project will build the system library drivers for the specific SOPC system.
This project inherits the name from the main software project and appends "_bsp" to that.

Initial content of the project
Since you chose the "blank" project template, there are no source files in the application project directory at this time. The BSP contains a directory of software drivers as well as a system.h header file, system initialization source code and other software infrastructure.
6.3 Add source code to the project

In Windows Explorer locate the project directory which contains a directory called “sw_source”. This directory contains an “inc” directory, “src” directory and “main.c” file.

You will copy these files and directories from Windows Explorer into the Eclipse software project directory, “DE0_Nano_led_control”.

Select the 3 objects and drag them over the “DE0_Nano_led_control” directory in the SBT window and drop the files onto the project folder. This should cause the source files to be physically copied into the file system location of the software project directory and register these source files within the Eclipse workspace so that they appear in the Project Explorer file listing.

Then drag and drop the simple_pwm_regs.h file (from the same “sw_source” directory) into the BSP project (DE_Nano_led_control_bsp). This file acts as the drive r for this device.

6.4 Configure Board Support Package

Configure the board support package to specify the properties of this software system by using the BSP Editor tool. These properties include what interface should be used for stdio and stderr messages, which memory should stack and heap be allocated in and whether an operating system or network stack should be included with this BSP.

Right click on the DE0_Nano_led_control_bsp project and select Nios II -> BSP Editor… from the right-click menu.

The software project provided in this lab does not make use of an operating system. All stdout, stdin and stderr messages will be directed to the jtag_uart. The auto-generated linker script will be used and the various linker sub-sections (Program memory, Read-only data memory, Read/write data memory) will be stored sdram. We will point the linker to place the heap and stacks in sdram.

In the “Common” settings view, change the following settings:

⇒ Select the sys_timer peripheral as the hardware for the sys_clk_timer .
⇒ Select the hires_timer peripheral as the hardware for the timestamp_timer .
⇒ Select sdram as the linker target for exception_stack_memory_region_name .
⇒ Select sdram as the linker target for interrupt_stack_memory_region_name .
Click on the **Linker Script** tab and change the following linker regions:

- Point the `.heap` linker section to the `sdram`
- Point the `.stack` linker section to `sdram`

⇒ Select **File** -> **Save** to save the board support package configuration to the `settings.bsp` file.
⇒ Click the **Generate** button to update the BSP.
⇒ When the generate has completed, select **File** -> **Exit** to close the BSP Editor.

### 6.5 Configure BSP Project Build Properties

In addition to the board support package settings configured using the BSP Editor, there are other compilation settings managed by the Eclipse environment such as compiler flags and optimization level.

- Right click on the `DE0_Nano_led_control_bsp` software project and select **Properties** from the right-click menu.
- On the left-hand menu, select the **Nios II BSP Properties** tab
- During compilation, the code may have various levels of optimization which is a tradeoff between code size and performance.
- Change the **Optimization level** setting to **Level 2**.
- Since our software does not make use of C++, **uncheck** “Support C++”.

![Properties for DE0_Nano_led_control_bsp](image)

Click **Apply**. Click **OK**.

### 6.6 Configure Application Project Build Properties

Just as you configured the optimization level for the BSP project, you should set the optimization level for the application software project “DE0_Nano_led_control” as well.

- Right click on the `DE0_Nano_led_control` software project and select **Properties** from the right-click menu.
- On the left-hand menu, select the **Nios II Application Properties** tab
- Change the **Optimization level** setting to **Level 2**.
- Click **Apply**. Click **OK**.

![Properties for DE0_Nano_led_control](image)
6.7 Build the software project

Right click the **DE0_Nano_led_control_bsp** software project and choose **Build Project** to build the board support package. When that build completes, right click the **DE0 Nano_led_control** application software project and choose **Build Project** to build the Nios II application.

These 2 steps will compile and build the associated board support package, then the actual application software project itself. The result of the compilation process will be an Executable and Linked Format file for the application, the (*.elf) file.

---

6.8 Run the software application on the target

To run any application on the target hardware, two images are needed

- The FPGA hardware image SDRAM Object File <.SOF>.
- The software executable the <.ELF>.

In the previous module you already downloaded the .SOF, so the FPGA is primed and ready to run the software application. Keeping the DE0 Nano kit still plugged into the USB port, you will download the application via the USB-JTAG link.

To run the software project on the Nios II processor:

- Right click on the software project directory and choose **Run As** and **Nios II Hardware**.

This will re-build the software project to create an up-to-date executable and then download the code into memory on our DE0 Nano hardware. The debugger resets the Nios II processor, and it executes the downloaded code.

You may receive a message that your target connection could not be determined, and a “Run Configuration” dialogue window will be presented to you. If the “Arrow-USB-Blaster” does not appear in the Connections lists, then click on “Refresh Connections” and then select “Arrow-USB-Blaster”. If more than one JTAG cable is shown in the list then be sure to select the “Arrow-USB-Blaster” connection. Then click the **Apply** button and then the **Run** button.

---

6.9 Interact with the Software Application

Once you have the DE0 Nano_led_control application running on the Nios II processor, you can interact with the demo by using your keyboard to control the program flow.

6.10 Edit the Application

You can optionally modify the led_util.c source file to change the software such that the counting LEDs are inverted.

Open the file led_util.c and locate the following subroutine:

```c
update_ledg( )
```

In the beginning of the subroutine add the following line:

```c
display_value = ~ display_value;
```

Once this is rebuilt, the application can be rerun to see the change in LEDs.

**CONGRATULATIONS!!**

You have just built the software application, downloaded it to the target and run the application on the target.
Taking the Next Step

After you have sufficiently familiarized yourself with the embedded system development flow, you may want to add a
SOPC Builder system to your design.

If you are starting from scratch, a good idea is to purchase a Nios II development kit, which comes with pre-generated
Nios II processor systems to accelerate your development flow as well as the Nios II IP license.

If you already have a working project then you can add the SOPC builder system to your top level as a stub or even
add your design to the SOPC Builder system as a custom component.

Either way you will find plenty of resources to get your job done on Altera’s embedded resources at
www.altera.com/embedded

Purchase an evaluation or development kit

Embedded Development Kit Resources

Get more information about the Nios II Processor

The Nios II Processor Reference Handbook

Get more information about the Nios II Software Development Tools

The Nios II Software Developers Handbook

Get more information about Embedded System Design

Embedded Design Guide

Get more information about SOPC Builder and Embedded IP Peripherals

SOPC Builder (Quartus II) Handbook
http://www.altera.com/literature/lit-sop.jsp

Get Ready made Nios II Processor System Design Examples and Software Applications

Nios II Design Examples page
http://www.altera.com/support/examples/nios2/exm-nios2.html

Get Free Online Tutorials or take an In person training course

Embedded Training Resources

Get support for your development by joining the Nios II User Community

Nios II User Community
http://www.altera.com/technology/embedded/community/emb-community.html

Get Technical Support from Altera

Embedded Technical Support
http://www.altera.com/technology/embedded/support/emb-support.html

For all resources visit www.altera.com/embedded