My First Nios II

Hardware
Create a new Quartus project.

Create a new Nios II system using the Qsys tool

1. From the Quartus menu, select **Tools | Qsys**

You will see the initial Qsys window with the clock component already added

Find the component that you want in the **Component Library** tab on the left and click **Add** to add it to your Nios II system.
**Add the On-Chip Memory**

1. On the Component Library tab (to the left of the System Contents tab), expand Memories and Memory Controllers, expand On-Chip, and then click On-Chip Memory (RAM or ROM).

2. Click Add. The On-Chip Memory (RAM or ROM) parameter editor appears.

3. In the Block type list, select Auto.

4. In the Total memory size box, type **20480** to specify a memory size of 20 KB.

5. Click Finish. You return to Qsys.
6. Click the **System Contents** tab. An instance of the on-chip memory appears in the system contents table.

7. In the **Name** column of the system contents table, right-click the on-chip memory and click **Rename**.

8. Type `onchip_mem` and press **Enter**.

   You must type these tutorial component names exactly as specified. Otherwise, the tutorial programs written for this Nios II system will fail in later steps.

9. In the **Connections** column, connect the `clk` port of the `clk_0` clock source to the `clk1` port of the **on-chip memory** by clicking the hollow dots on the connection line. The dots become solid indicating the ports are connected.

10. Connect the `clk_reset` port of the `clk_0` clock source to the `reset1` port of the on-chip memory.
Add the Nios II Processor Core

This is the CPU.

1. On the Component Library tab, expand Embedded Processors, and then click Nios II Processor.
2. Click Add. The Nios II Processor parameter editor appears.
3. Select the Core Nios II tab.
4. For the Nios II Core, select Nios II/s.
5. In the Hardware multiplication type list, select None.
6. Uncheck Hardware divide.
7. Click Finish.
8. In the Name column, right-click the Nios II processor and click Rename.
9. Type cpu and press Enter.
10. In the Connections column, connect the clk port of the clk_0 clock source to the clk port of the Nios II processor.
11. Connect the clk_reset port of the clk_0 clock source to the reset_n port of the Nios II processor.
12. Connect the s1 port of the on-chip memory to both the data_master port and instruction_master port of the Nios II processor.

13. Double-click the Nios II processor row of the system contents table to reopen the Nios II Processor parameter editor.
14. Under Reset Vector, select onchip_mem.s1 in the Reset vector memory list.
15. The Reset vector offset and the Reset vector should both be 0x00000000.
16. Under Exception Vector, select onchip_mem.s1 in the Exception vector memory list.
17. The Exception vector offset and the Exception vector should both be 0x00000020.
18. Click Finish.
## Select a Nios II Core

<table>
<thead>
<tr>
<th></th>
<th>Nios II/e</th>
<th>Nios II/s</th>
<th>Nios II/f</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nios II</strong></td>
<td>RISC 32-bit</td>
<td>RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide</td>
<td>RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dynamic Branch Prediction</td>
</tr>
<tr>
<td>Memory Usage (e.g. Stratix IV)</td>
<td>Two M6Ks (or equiv.)</td>
<td>Two M6Ks + cache</td>
<td>Three M6Ks + cache</td>
</tr>
</tbody>
</table>

### Hardware Arithmetic Operation

- **Hardware multiplication type:** None
- **Hardware divide**

### Reset Vector

- **Reset vector memory:** onchip_mem.s1
- **Reset vector offset:** 0x00000000
- **Reset vector:** 0x00000000

### Exception Vector

- **Exception vector memory:** onchip_mem.s1
- **Exception vector offset:** 0x00000020
- **Exception vector:** 0x00000020
**Add the JTAG UART**

The JTAG UART provides a convenient way to communicate character data with the Nios II processor through the USB-Blaster download cable.

1. On the Component Library tab, expand *Interface Protocols*, expand *Serial*, and then click JTAG UART.
2. Click Finish.
3. Rename the component to `jtag_uart`.
4. Connect the `clk` port of the `clk_0` clock source to the `clk` port of the `jtag_uart`.
5. Connect the `clk_reset` port of the `clk_0` clock source to the `reset` port of the `jtag_uart`.
6. Connect the `data_master` port of the Nios II processor to the `avalon_jtag_slave` port of the `jtag_uart`.

The instruction_master port of the Nios II processor does not connect to the JTAG UART because the JTAG UART is not a memory device and cannot send instructions to the Nios II processor.

7. In the **IRQ** column for the `avalon_jtag_slave` row, connect the `cpu` to the `jtag_uart` by clicking on the hollow dot.

IRQ’s are CPU interrupts. The Nios II HAL interprets low IRQ values as having higher priority.

8. A textbox appears. Type in the number **16** and press **Enter** to assign a new IRQ value.
**Add the Interval Timer**

Most control systems use a timer component to enable precise calculation of time. To provide a periodic system clock tick, the Nios II HAL requires a timer.

1. On the Component Library tab, expand **Peripherals**, expand **Microcontroller Peripherals**, and then click **Interval Timer**.
2. Click Add.
3. In the **Presets** list, select **Full-featured**.
4. Click Finish.
5. Rename the interval timer to **sys_clk_timer**.
6. Connect the clock.
7. Connect the clock reset.
8. Connect the **data_master** port of the Nios II processor to the **s1** port of the interval timer.
9. In the **IRQ** column, connect the **cpu** to the **sys_clk_timer** by clicking on the hollow dot.
10. A textbox appears. Type in the number **1** and press **Enter** to assign a new IRQ value.

The Nios II HAL interprets low IRQ values as higher priority. The timer component must have the highest IRQ priority to maintain the accuracy of the system clock tick.

---

**Add the System ID Peripheral**

The system ID peripheral safeguards against accidentally downloading software compiled for a different Nios II system. If the system includes the system ID peripheral, the Nios II SBT for Eclipse can prevent you from downloading programs compiled for a different system.

1. On the Component Library tab, expand **Peripherals**, expand **Debug and Performance**, and then click **System ID Peripheral**.
2. Click Add.
3. Change the 32 bit System ID to any unique hex number you like.
4. Click Finish.
5. Rename the System ID Peripheral to **sysid**.
6. Connect the clock.
7. Connect the clock reset.
8. Connect the **data_master** port of the Nios II processor to the **control_slave** port of the System ID Peripheral.
Add the PIO

PIO signals provide an easy method for Nios II processor systems to receive input stimuli and drive output signals. Complex control applications might use hundreds of PIO signals which the Nios II processor can monitor. This design example uses eight PIO signals to drive LEDs on the board.

1. On the Component Library tab, expand Peripherals, expand Microcontroller Peripherals, and then click PIO (Parallel I/O).
2. Click Add.
   The parameter editor defaults to an 8-bit output-only PIO, which is what we want.
3. Click Finish.
4. Rename the PIO to led_pio.
5. Connect the clock.
6. Connect the clock reset.
7. Connect the data_master port of the Nios II processor to the s1 port of the PIO.
8. In the external_connection row of the PIO component, double-click Double-click to export in the Export column to export the PIO ports. Press Enter to accept the default name. This will create connections in the Nios II cpu symbol for you to connect to your output pins.
Specify Base Addresses and Interrupt Request Priorities to all of the components

1. In the Qsys main menu, select System | Assign Base Addresses to make Qsys assign functional base addresses to each component in the system. Values in the Base and End columns might change, reflecting the addresses that Qsys reassigned.
2. In the Qsys main menu, select System | Assign Interrupt Numbers.

Complete Nios II System

The following is the complete Nios II system.

[Diagram of Nios II system]

Generate the Nios II System

1. Click on the Generation tab.
2. Click on the Generate button.
3. When the Save changes dialog box appears, click Save.
4. Type in first_nios2_system in the File name box.
5. Click Save.
6. The system generation should complete with 0 errors and 0 warnings.
7. Select File | Exit to close the Qsys window.
Making changes to the Nios II System

If you need to make changes to your Nios II system, open Qsys again and select your Nios system `first_nios2_system.qsys` in the Open dialog, then click Open.
**Add IP Variation File**

You need to add the Nios II system IP File (.qip) to the your Quartus II project.

1. Under the Quartus main menu, select **Assignments | Settings**. The Settings dialog box appears.

2. Select **Files**.
3. Next to File name, click the browse (...) button.
4. In the **Files of type** list, select **Script Files (*.tcl, *.sdc, *.qip)**.
5. Browse to locate `first_nios2_system/synthesis/first_nios2_system.qip` and click **Open** to select the file.
6. Click **Add** to include `first_nios2_system.qip` in the project.
7. Click **OK** to close the Settings dialog box.
Integrate your Nios II System into your Quartus Project

1. Open a new Block Diagram/Schematic file.
2. Click on the Symbol Tool to bring up the Symbol window.
3. Under the Project folder, you will see your first_nios2_system symbol. Select that to insert the component into your schematic file.

4. Connect the clock signal to an input pin and map it to the 50 MHz clock source.
5. Connect the reset signal to VCC.
6. Connect the led_pio signal to an output pin and map it to the seven green leds.
**Synthesize and upload your Nios to the DE1 board**

1. Synthesize your Nios II design.
2. Program (upload) your Nios II design onto the DE1 board.
3. Because of the time limited open source licensing of the Nios II, you will see the following window. Click OK to accept it.

![Quartus II window](image)

4. After you have successfully programmed your Nios II onto the DE1 board, you need to keep both the programmer window and the following OpenCore Plus Status window opened in order for your Nios II system to run on the DE1 board. This is because of the time limited open source licensing of the Nios II.

![OpenCore Plus Status window](image)

Note that the programmer window and/or the OpenCore Plus Status window may be hidden behind the Quartus window. So don’t keep opening up another programmer window just because you don’t see it right away!
Software
Now we are ready to run our first Hello World program written in C on our Nios II system.

1. Start the Nios II Eclipse program
2. In the Workspace window, browse to your Quartus project folder and add a subdirectory called Software.
3. Click OK
Create a new C program

1. From the Eclipse menu, select File | New | Nios II Application and BSP from Template
2. In the Nios II Application and BSP from Template window, do the following:
   a. For the SOPC Information File name, select your first_nios2_system.sopcinfo
   b. For the Project name, type in hello_world
   c. For the Project template, select Hello World
3. Click Finish
Tell the compiler to reduce the size of the code for small memory

1. In the Eclipse Project Explorer window, right-click on hello_world_bsp (your project’s bsp folder)
2. Select Properties in the pop-up menu
3. Select Nios II BSP Properties on the left
4. Check Reduced device drivers and Small C library
5. Uncheck all the others
6. Click OK
Compile and run your C program on the Nios II hardware

1. In the Eclipse Project Explorer window, right-click on hello_world (your project’s source folder)
2. Select Run As in the pop-up menu
3. Select Nios II Hardware in the pop-up menu

You will see the following trace as your program is being compiled and uploaded to your Nios hardware

Using cable "USB-Blaster [USB-0]", device 1, instance 0x00
Processor is already paused
Reading System ID at address 0x01111080: verified
Initializing CPU cache (if present)
OK

Downloading 00800000 ( 0%)
Downloading 00801FEC (90%)
Downloading 01080000 (99%)
Downloaded 9KB in 0.1s

Verifying 00800000 ( 0%)
Verifying 00801FEC (90%)
Verifying 01080000 (99%)
Verified OK
Starting processor at address 0x00800194

If you get error messages, then see the solutions at the end of this document.

After the upload, the program will run automatically and the words “Hello from Nios II!” should be printed in the Nios II Console output
Each time after modifying the Nios II hardware, you need to update the system.h file for your software project.

1. Right-click on hello_world_bsp (your project’s bsp folder)
2. Select Nios II from the drop-down menu
3. Select Generate BSP
A larger Nios II system with external SRAM, SDRAM, seven-segment LEDs, button PIOs, and PS2 keyboard.
Error message:

![Problem Occurred](image)

'Launching count_binary Nios II Hardware configuration’ has encountered a problem.
Downloading ELF Process failed

OK Details >>

Cause:

Your Nios II hardware has problems. Most likely the CPU’s **Reset Vector** and/or **Exception Vector** are pointing to invalid or bad memory.
Error message:

![Image of Nios II HardwareLauncher with error message]

Cause:

Your memory (most likely your onchip memory) is too small to fit the program.

Solution 1: Make your program smaller

4. In the Project Explorer window, right-click on project’s bsp (hello_world_bsp)
5. Select Properties in the pop-up menu
6. Select Nios II BSP Properties on the left
7. Check Reduced device drivers and Small C library. Uncheck all the others

![Image of Nios II BSP Properties dialog]

8. Click OK
Solution 2: Use SRAM or SDRAM. This of course is only possible if you have SRAM or SDRAM in your Nios system.

1. In the Project Explorer window, right-click on project’s bsp (hello_world_bsp)
2. Select Properties in the pop-up menu
3. Select Nios II BSP Properties on the left
4. Click on the BSP Editor... button
5. Select the Linker Script tab at the top
6. For all the Linker Section Names, select sram or sdram in the Linker Region Name

7. Click Generate
8. Click Exit