# Using NIOS 2 Embedded Design Suite 10

Quick Start Guide

Embedded System Course

LAP – IC – EPFL – 2010

Version 0.1 (Preliminary)

Cagri Onal, René Beuchat

## 1 Installation and documentation

*Main information in this document has been found on:*

http:\\www.altera.com

*This guide has been prepared to help students following the Embedded System Course in I&C by Cagri Onal at EPFL. NIOS2 EDS 10 is used to create, download and debug embedded software programs for the hardware systems created by Quartus II from Altera.*

*Copy of the installation files for NIOS2 EDS 10 can be found at LAP for personal installation:*

| |
|---|
| *\\lapsrv1\distribution\Altera\Tools_For_Windows\To_install_QuartusII_10_0\ 10.0_nios2eds_windows_rev2.exe* |
| *\\lapsrv1\distribution\Altera\Tools_For_Windows\To_install_QuartusII_10_0\ 10.0sp1_nios2eds_windows.exe* |

(or you can follow http:\\www.altera.com to download the install files after registration).

# 2 Launching NIOS2 EDS

NIOS2 EDS software can be launched by following **"Start->All Programs->Altera->Nios II EDS 10.0->Nios II 10.0 Software Build Tools for Eclipse"**

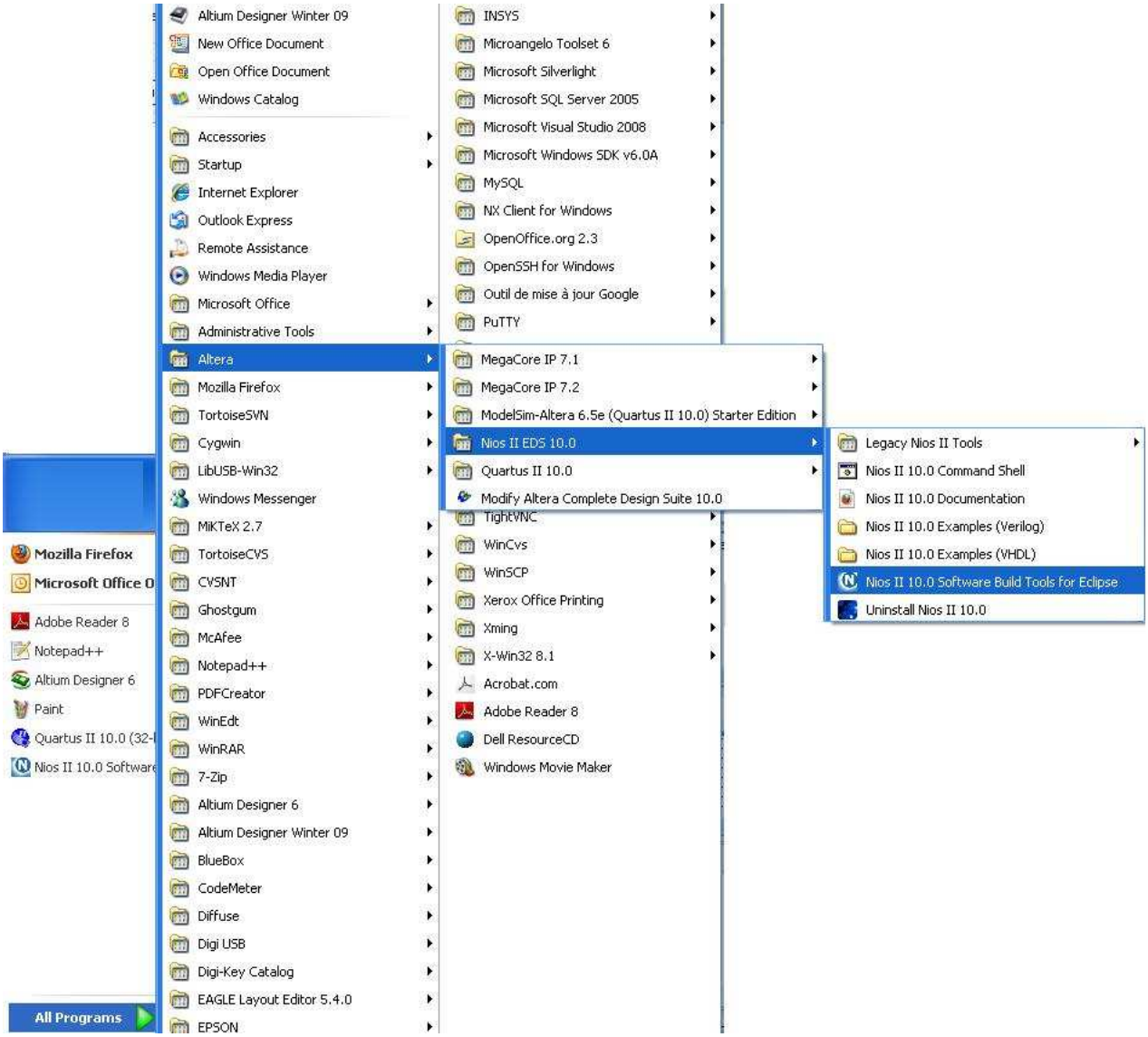


Fig. 1. Launching NIOS 2 EDS 10.0 I

**OR**

by clicking the **"Nios II Software Build Tools for Eclipse"** button under the **System Generation** tab of **Altera SOPC Builder**.


Fig. 2. Launching NIOS 2 EDS 10.0 II

## 2.1 Select a workspace

A prompt will appear and ask for a workspace location.

**ATTENTION:** Do not choose the folder that has your hardware system files created by Quartus. Instead, create a new dummy folder, **e.g. Z:\niosworkspace** and select it as the workspace location.

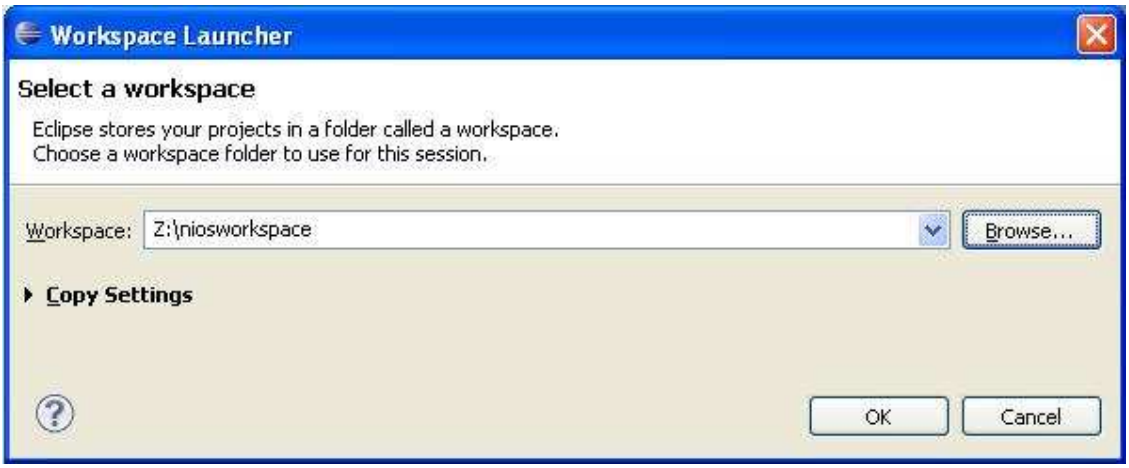


Fig. 3. Select a Workspace

## 2.2 Open NIOS II Perspective

Nios 2 EDS is an eclipse-based software tool. There are several perspectives such as **C/C++ and Nios II** for several purposes. For instance, one can write general purpose programs using **C/C++ perspective,** compile and run them on a computer. However, since the aim in this course is to create and download embedded software, the perspective to be activated will be always **Nios II**. It should be automatically selected on the top right corner of the empty Nios II page after launching Nios 2 EDS.

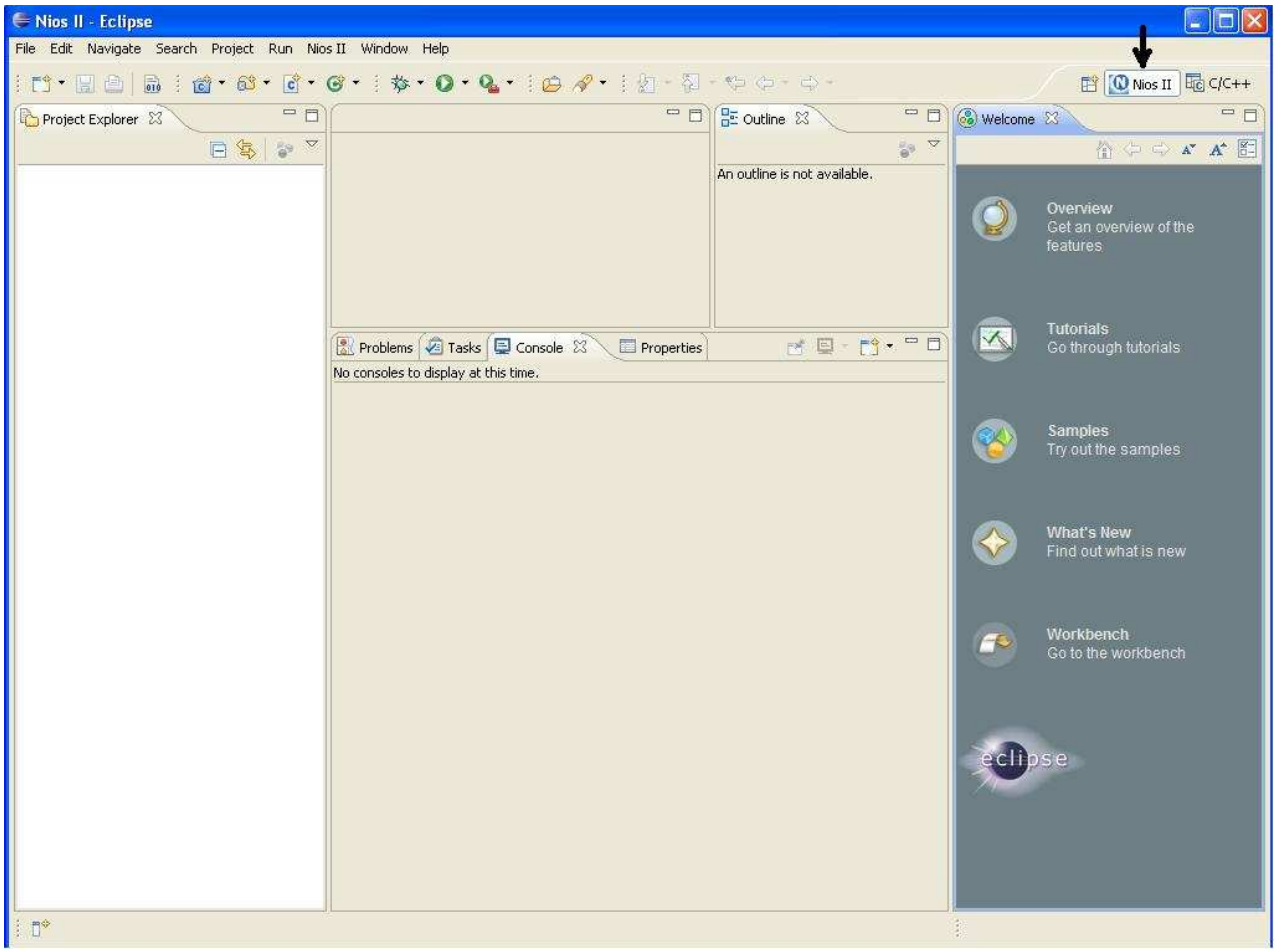


Fig. 4. Nios II Perspective

If **Nios II perspective** does not appear, or it is closed by mistake, it can be opened by following **"Window->Open Perspective->Other…"** and selecting NIOS II from the **"Open Perspective"** list.

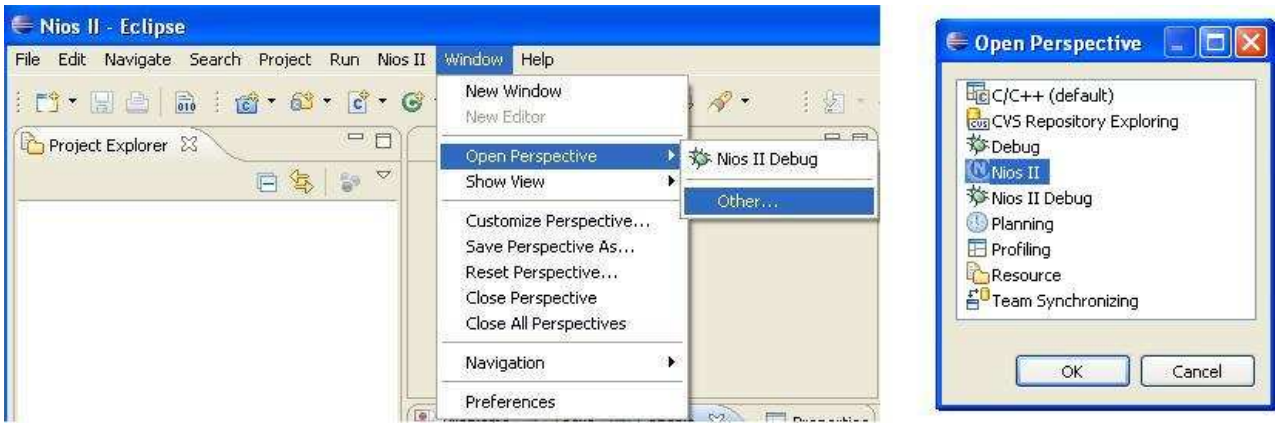


Fig. 5. Open Perspective

## 2.3 Creating a NIOS II Application and Board Support Package (BSP)

Board Support Package (BSP) has the information to establish the connection between the software application and the hardware system. This information is actually formed by the implementation of a Hardware Abstraction Layer (HAL) and the device drivers.

To create new NIOS II Application and BSP, select menu **"File->New->Nios II Application and BSP from Template"**

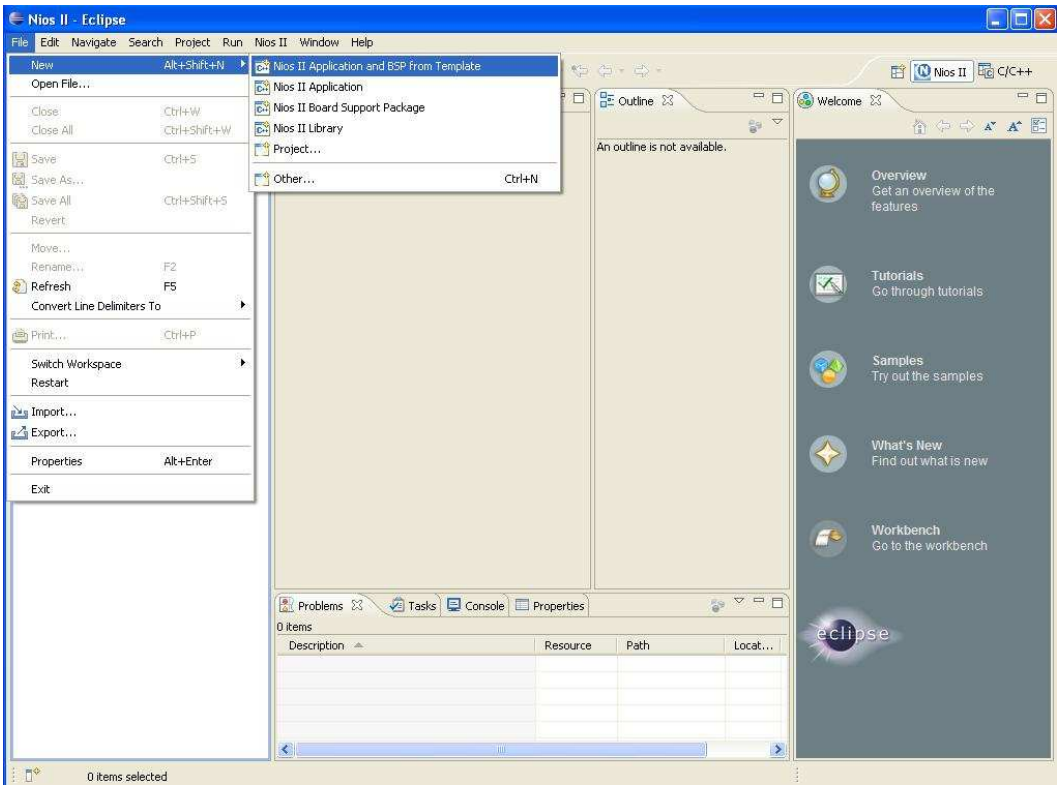


Fig. 6. Creating a NIOS II Application and Board Support Package

The following dialog will appear.

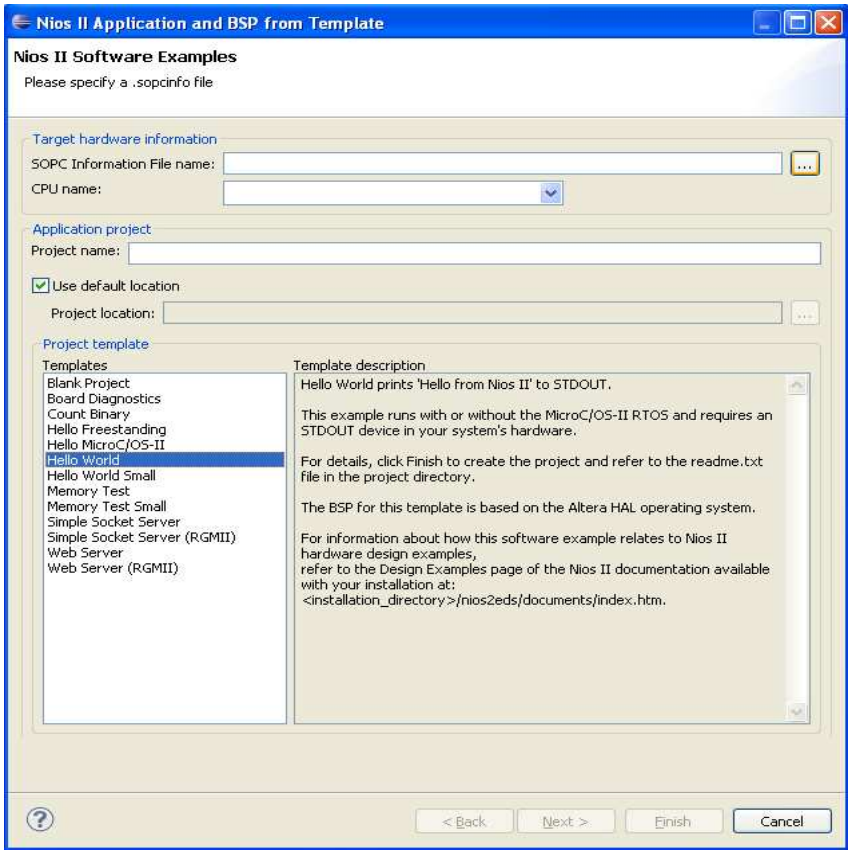


Fig. 7. Empty Creating NIOS II Application dialog

Then

1) Locate SOPC Information File name (specify a .sopcinfo file). I.e. browse for the folder that has the hardware system files created by Quartus.

2) Choose the CPU name; there will be one option in single CPU hardware systems and automatically selected. If more than one CPU available, choose the one for which the application is to be written.

3) Type the name for the application project and keep the **"Use default location"** box checked.

4) Choose Project template as **"Hello World Small"**. This is required if an On-Chip Memory smaller than 64K is used as program memory. Otherwise, using **"Hello World"** instead will cause errors while building the project since it will not fit in such memory.

The dialog will finally appear as the following:

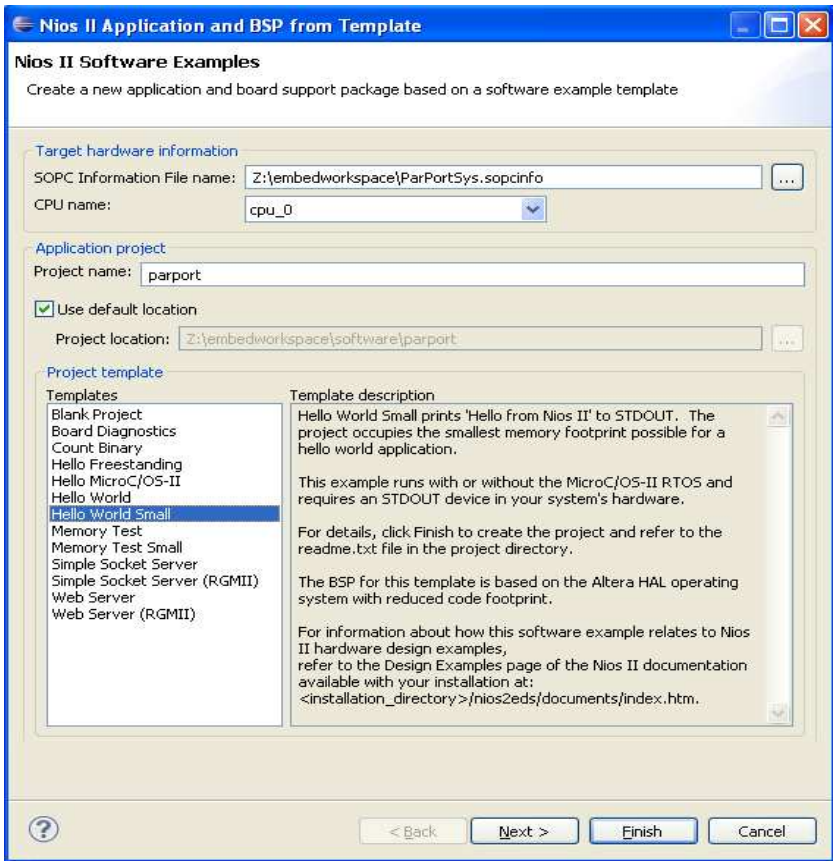


Fig. 8. Creating NIOS II Application dialog I

Then press **"Next".** In the next dialog, it is possible to select an existing BSP project from the workspace, but there is no need to make any changes for this tutorial. Thus, click **"Finish"** to generate the necessary files and folders.

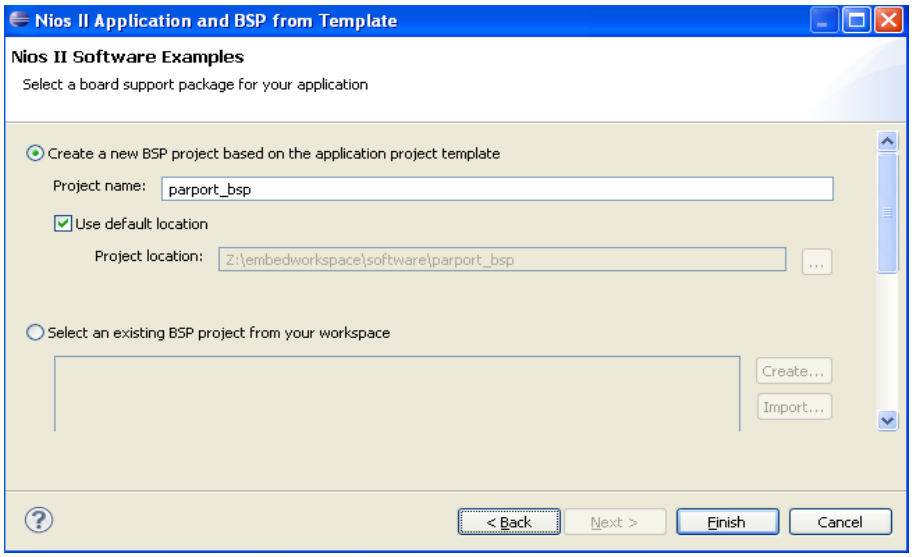


Fig. 9. Creating NIOS II Application dialog II

If everything works fine, the project and its corresponding BSP will be created successfully. Now let's **build** the project for the first time. **Right click** the application project and select **Build Project** from the opening menu. This will **first build the BSP project and then the application project**. Thus there is no need to rebuild the BSP project.

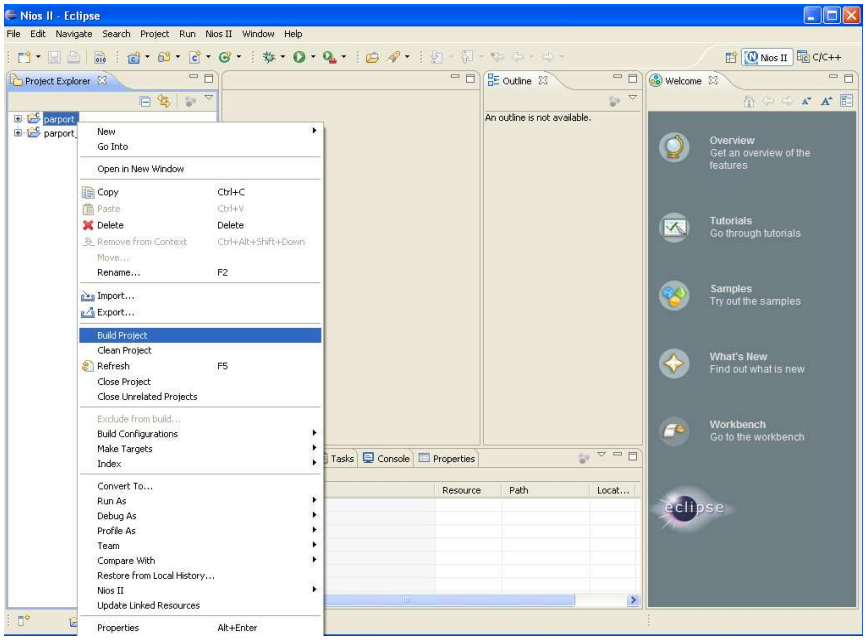


Fig. 10. Build Project

Now open the file **"hello_world_small.c"** under the application project folder by double clicking this file.

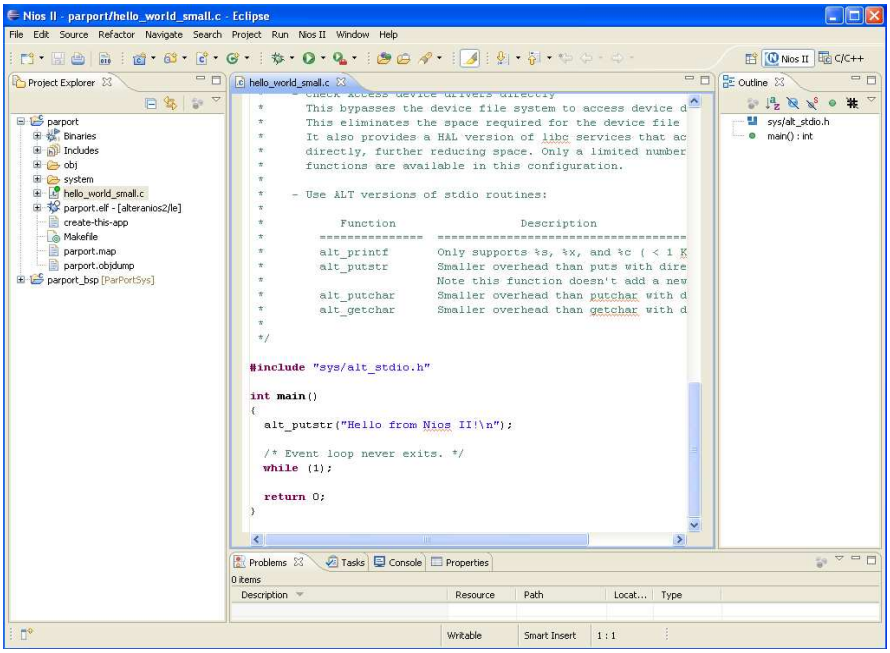


Fig. 11. Open **"hello_world_small.c"**

Add the following lines, save the file “hello_world_small.c” by **CTRL+S**, rebuild the application project.

**#include “system.h”**
**#include “io.h”**

```
#include "sys/alt_stdio.h"
#include "system.h"
#include "io.h"

int main()
{
  alt_putstr("Hello from Nios II!\n");

  /* Event loop never exits. */
  while (1);

  return 0;
}
```

Fig. 12. **“hello_world_small.c”** with **“system.h”** and **“io.h”** I

After rebuilding, **“system.h”** and **“io.h”** will be accessible on the right under **Outline** tab.

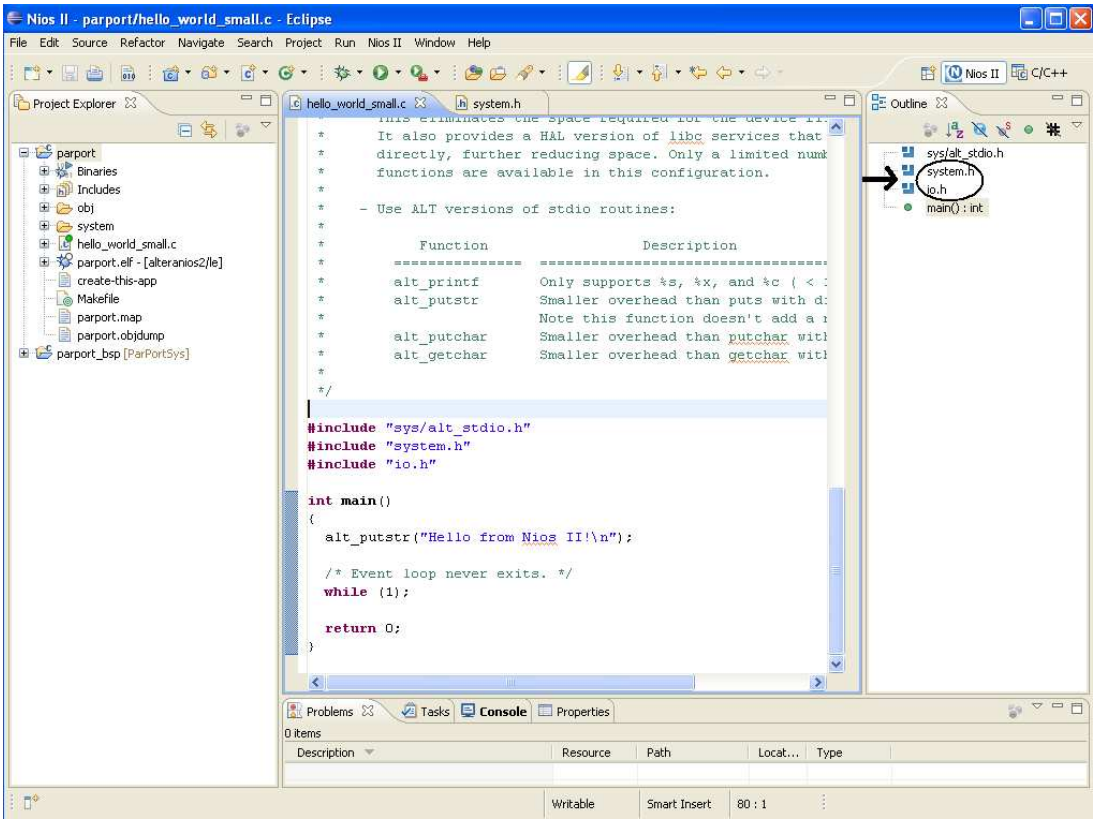


Fig. 13. **“hello_world_small.c”** with **“system.h”** and **“io.h”** II

**"system.h"** is the header file which has the macros defined for supplying the properties of the components in the hardware system to the user. Double click **"system.h"** under **Outline** tab to investigate the file.

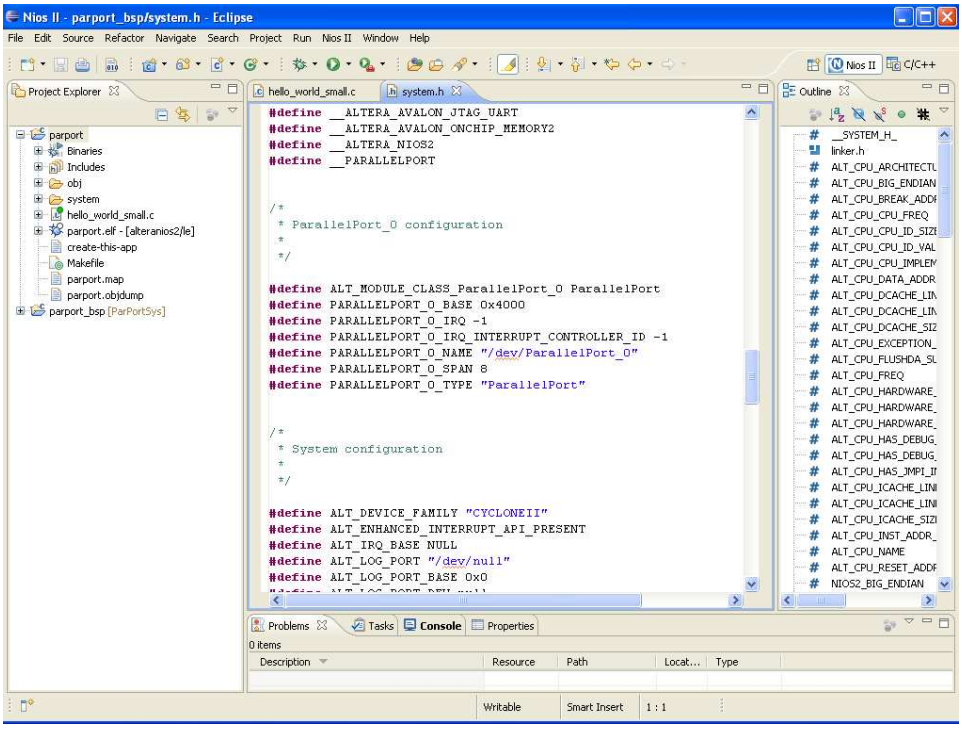


Fig. 14. **"system.h"**

**"io.h"** is the header file which has the macros of functions letting the user to access the Avalon Bus through the NIOS CPU by writing to or reading from the Avalon Bus in different modes. Double click **"io.h"** under **Outline** tab to investigate the file.

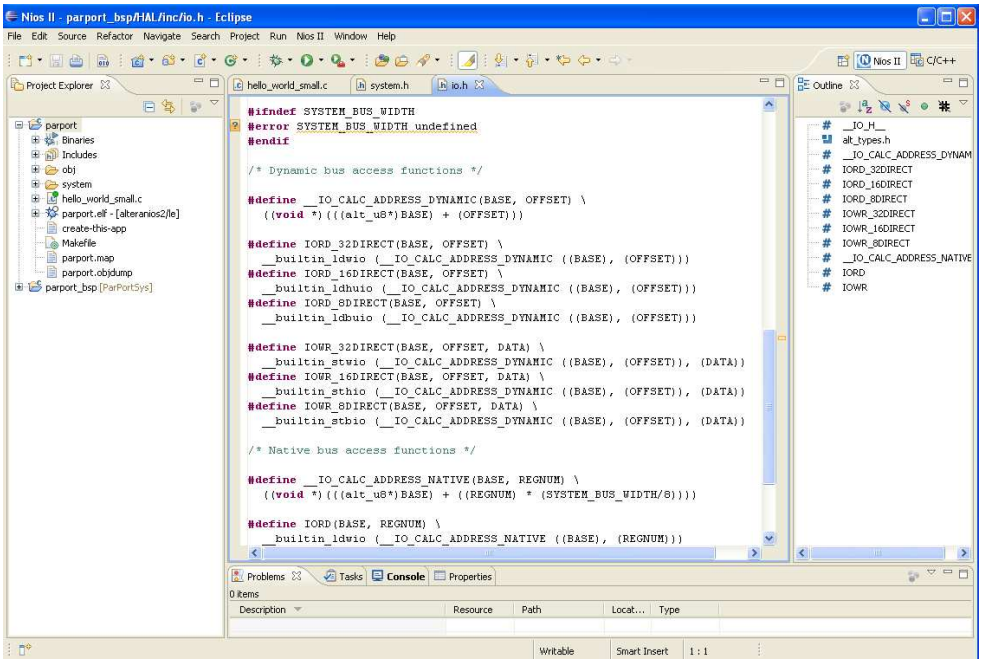


Fig. 15. **"io.h"**

# 3 Example program to start

Initialization software to test the parallel port is given as follows:

```
/*
 *       alt_printf       Only supports %s, %x, and %c ( < 1 Kbyte)
 */
#include "sys/alt_stdio.h"
#include "system.h"
#include "io.h"

#define IREGDIR  0  //Change the values if your register map is different than
here
#define IREGPIN  1
#define IREGPORT 2
#define MODE_ALL_OUTPUT 0xFF
#define MODE_ALL_INPUT  0X00

void init()
{
      volatile unsigned int k;
      while(1)
      {
      IOWR_8DIRECT(PARALLELPORT_0_BASE,IREGDIR,MODE_ALL_OUTPUT);
            //Select Parport as output
      alt_printf("iRegDir=%x\n",IORD_8DIRECT(PARALLELPORT_0_BASE,IREGDIR));
      //Read iRegDir to check whether it is written correct
      IOWR_8DIRECT(PARALLELPORT_0_BASE,IREGPORT,0x9b);
            //Write Parport 0x9b as the output value
      alt_printf("iRegPort=%x\n",IORD_8DIRECT(PARALLELPORT_0_BASE,IREGPORT));
      //Read iRegPort to check whether it is written correct

            //Switch LEDS should give 0x9b, observe it
      for(k=0;k<4000000;k++); //software delay

      IOWR_8DIRECT(PARALLELPORT_0_BASE,IREGDIR,MODE_ALL_INPUT);
            //Select Parport as input
      alt_printf("iRegDir=%x\n",IORD_8DIRECT(PARALLELPORT_0_BASE,IREGDIR));
      //Read iRegDir to check whether it is written correct

      //Change the input value to a value different than 0x9b by changing switch
positions
      alt_printf("iRegPin=%x\n",IORD_8DIRECT(PARALLELPORT_0_BASE,IREGPIN));
      //Read iRegPin to take the input Parport value

      for(k=0;k<4000000;k++); //software delay
      }
}
int main()
{
  alt_putstr("Hello from Nios II!\n");
  init();
  /* Event loop never exits. */
  while (1);
  return 0;
}
```

Fig. 16. Example Program

Note that the initialization software does similar things as the test bench provided in Simulation with ModelSim (vers. 0.5) document. There is actually a correspondence between the functions in test bench and the macro functions in the software.

| Test Bench | Software |
|---|---|
| WrBus() | IOWR*(BASE, REGNUM, DATA) |
| RdBus() | IORD*(BASE, REGNUM) |

Fig. 17. Correspondence table

Therefore, after a correct timing simulation with Modelsim, it is easy to write the corresponding software code to be downloaded and executed on the hardware system with the corresponding macro functions.

* It might be Dynamic Access or Native Access functions, check **"io.h"** to decide which one is going to be used with the corresponding hardware.

# 4 Run/Debug NIOS 2 project

## 4.1 Create a Run Configuration

Since the methodology to create configurations for both running and debugging the project is the same, below it is only given how to **create a Run Configuration.** Either select menu **"Run->Run Configurations…"** or press the arrow next to **RUN button** and choose **"Run Configurations…"**

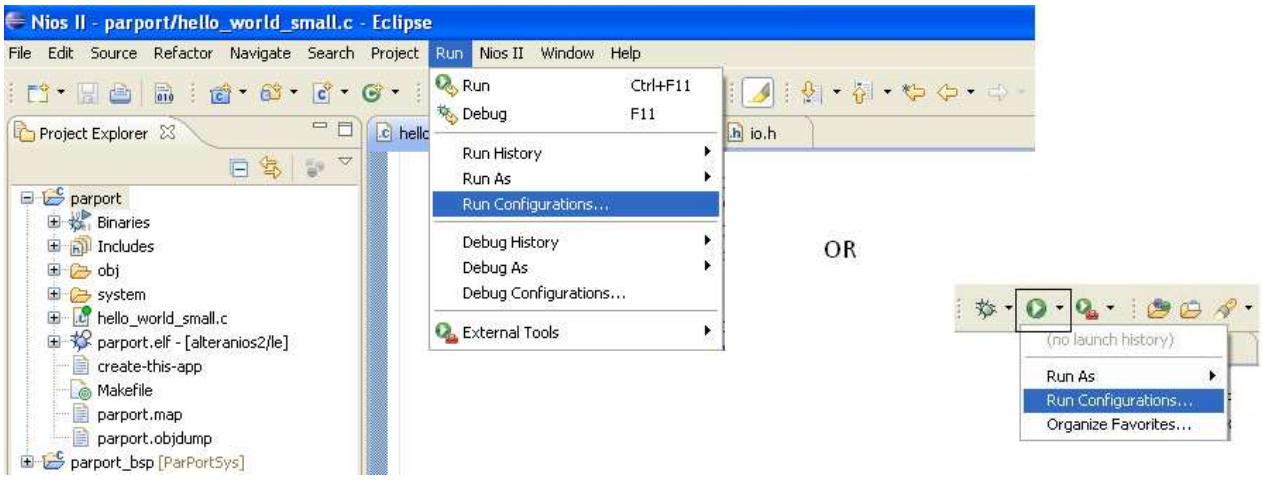


Fig. 18. Create a Run Configuration

Run Configurations window will open.

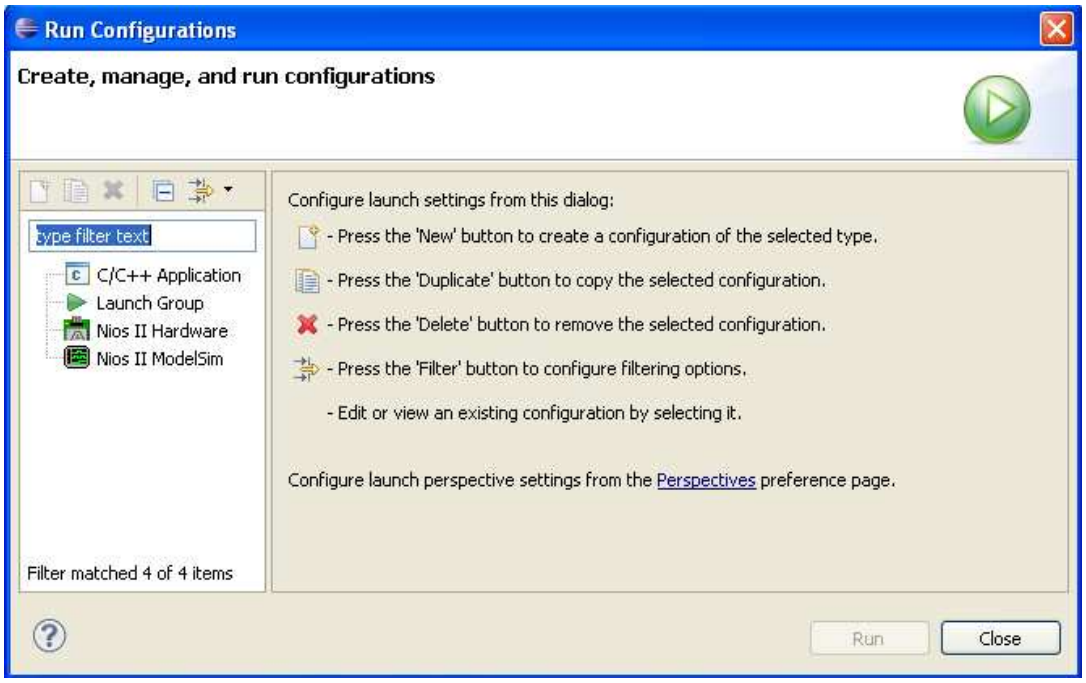


Fig. 19. Run Configurations window

Choose Nios II Hardware by double clicking. On the **Project** tab, modify the configuration name and select the project for which the configuration is created. Press **Apply** and move to tab **Target Connection**.

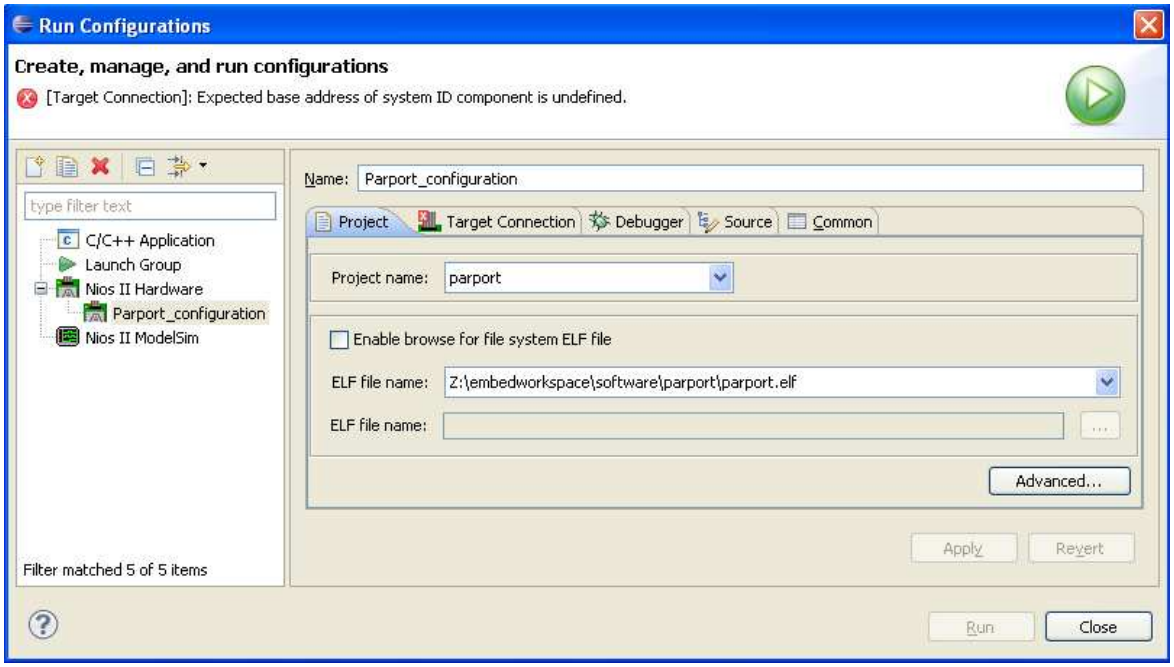


Fig. 20. Settings under Project tab

Enlarge the window to see the **Refresh Connections** button. Press this button to list the connections. Normally, USB-Blaster will be shown on the list if the related driver is installed successfully. If not, check the cable connected properly. If the problem persists, try to re-install the driver for USB-Blaster. The driver can be found in the following folder: **C:\altera\10.0\quartus\drivers\usb-blaster**

Check the box **Ignore mismatched system ID.**
Check the box **Ignore mismatched system timestamp.**

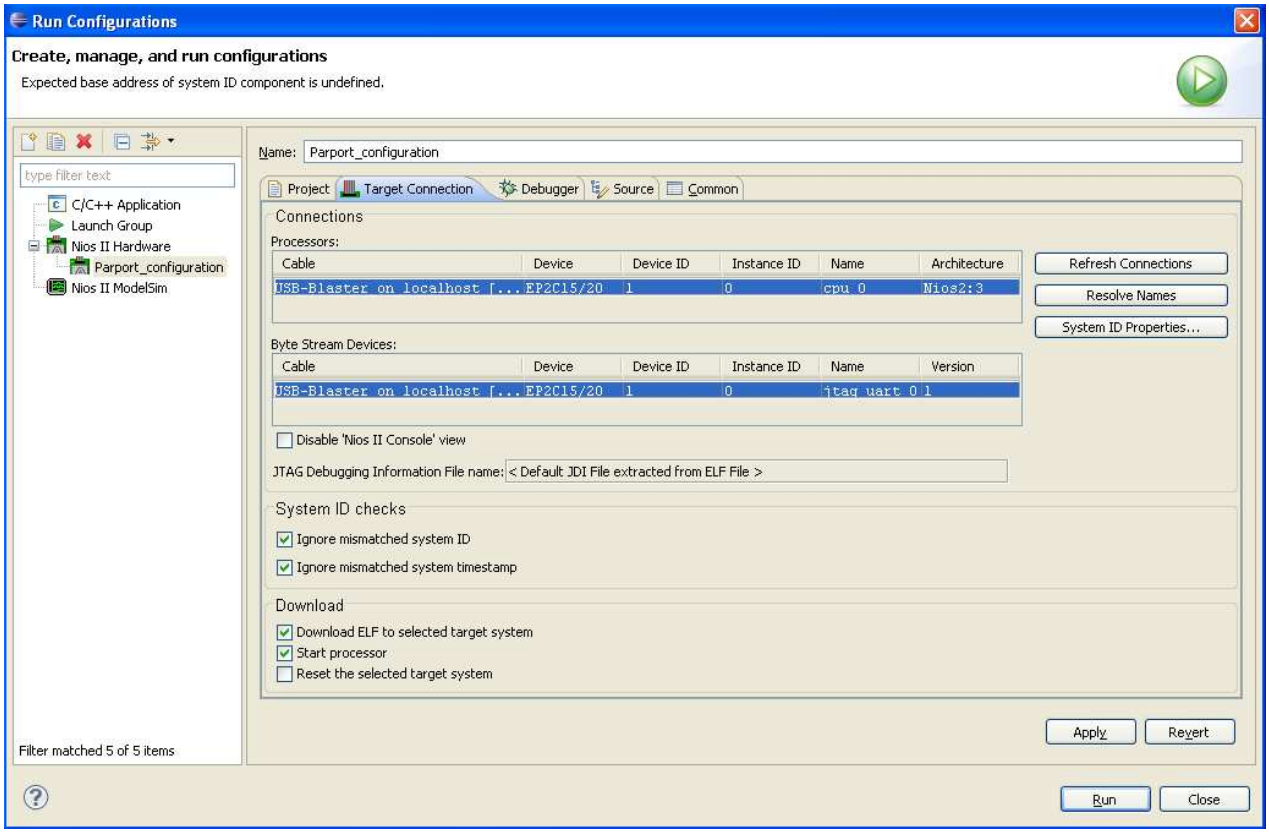


Fig. 21. Settings under Target Connection tab

Then press **Apply** and then **Run**. Thus, the program will be downloaded to the FPGA and executed for the first time. Note that there is no need to create configurations every time the software is modified. Just press the arrow next to **Run Button** and choose the previously created configuration.

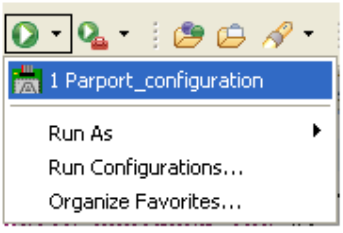


Fig. 22. Running the Configuration

## 4.2 Nios Debug Perspective

To debug the project, press the arrow next to **Debug Button** and choose the previously created configuration.

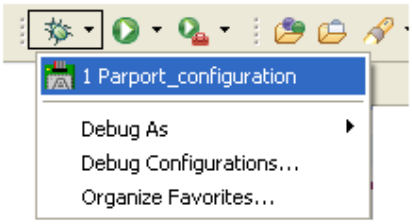


Fig. 23. Debugging the Configuration

Then the Nios Debug Perspective will open. On this perspective, the user can run or stop execution and observe variables, registers and the memory at the breakpoints.

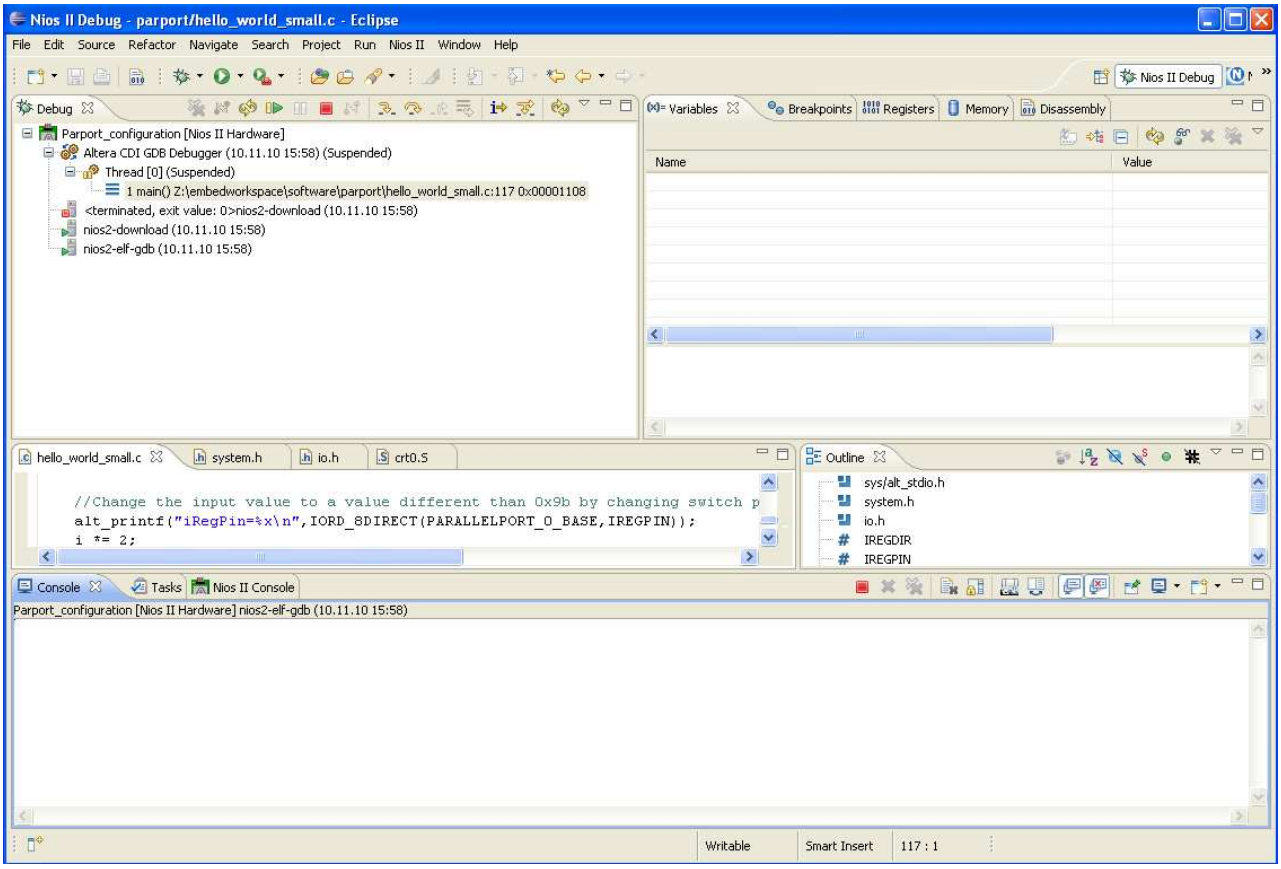


Fig. 24. Nios Debug Perspective

Summary

# List of Figures