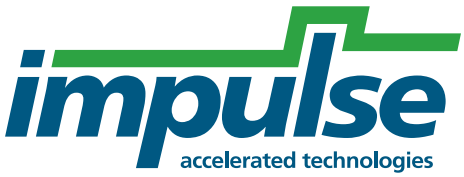


Application Note

Using Impulse C with the Visual Studio 2008 IDE

Michael Kreeger
Impulse Accelerated Technologies, Inc.

Overview

This application note describes how to integrate the Impulse CoDeveloper software and hardware design tools into a Visual Studio 2008 workflow.

Impulse C is based on ANSI C, allowing you to use standard C development tools to debug your application and model hardware-hardware and hardware-software interactions. In effect, the standard C compiler and debugger become a hardware/software behavioral simulator, allowing you to model parallel processes and observe the movement of data from one process to another, before generating any hardware. These software simulation methods are described in detail in the Impulse C User's Guide.

This application note describes how you can Microsoft Visual Studio for all of your Impulse C design processing, including both software simulation and hardware generation. The benefit of doing this is that you only need to use one IDE platform (Visual Studio) for both software debugging and hardware generation.

This document assumes that you are already somewhat familiar with the Impulse CoDeveloper design flow. If you are not familiar with this design flow, please read the introductory tutorials provided with Impulse CoDeveloper, in the Help and Support section of the CoDeveloper Start Page.

Requirements

- Microsoft Visual Studio 2008
- CoDeveloper 3.20.b.6 or above.
- Impulse Visual Studio Support installer (provided with CoDeveloper)

Step 1: Install the Impulse Visual Studio support files

The Visual Studio plugin for Impulse is installed during CoDeveloper Installation. Select Microsoft Visual C++ Plugin.

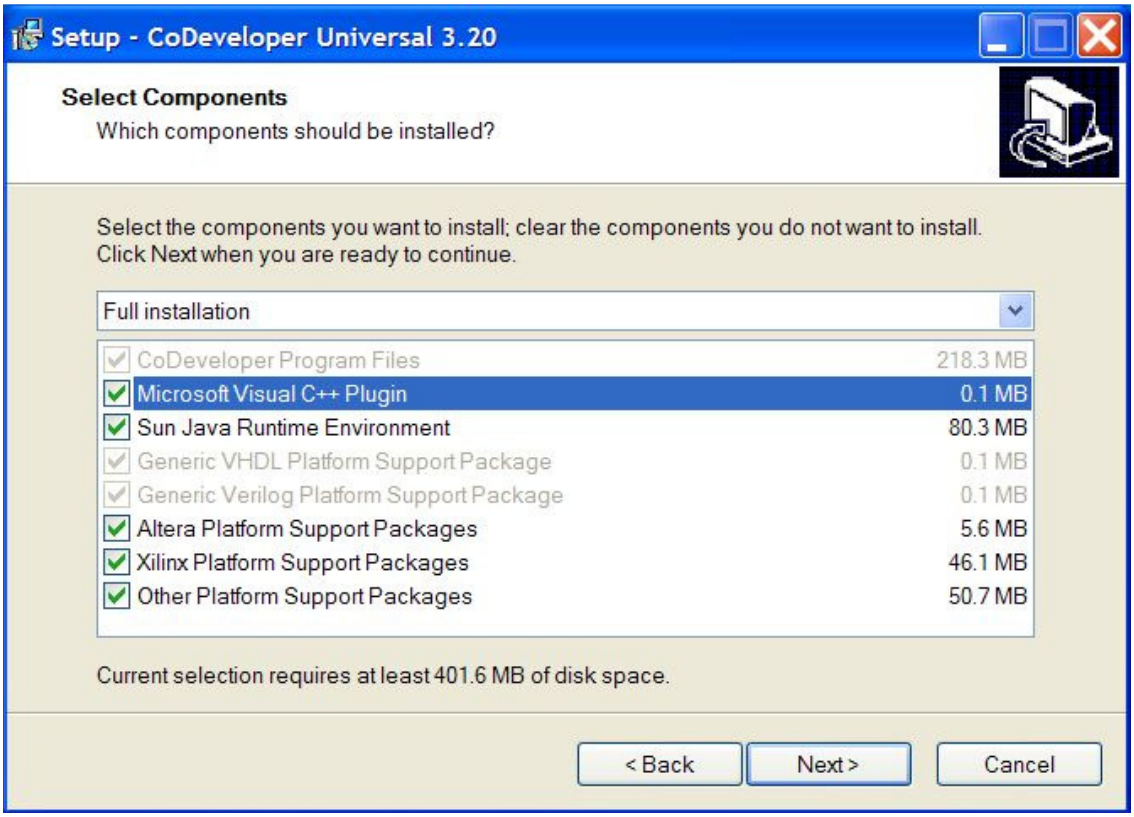
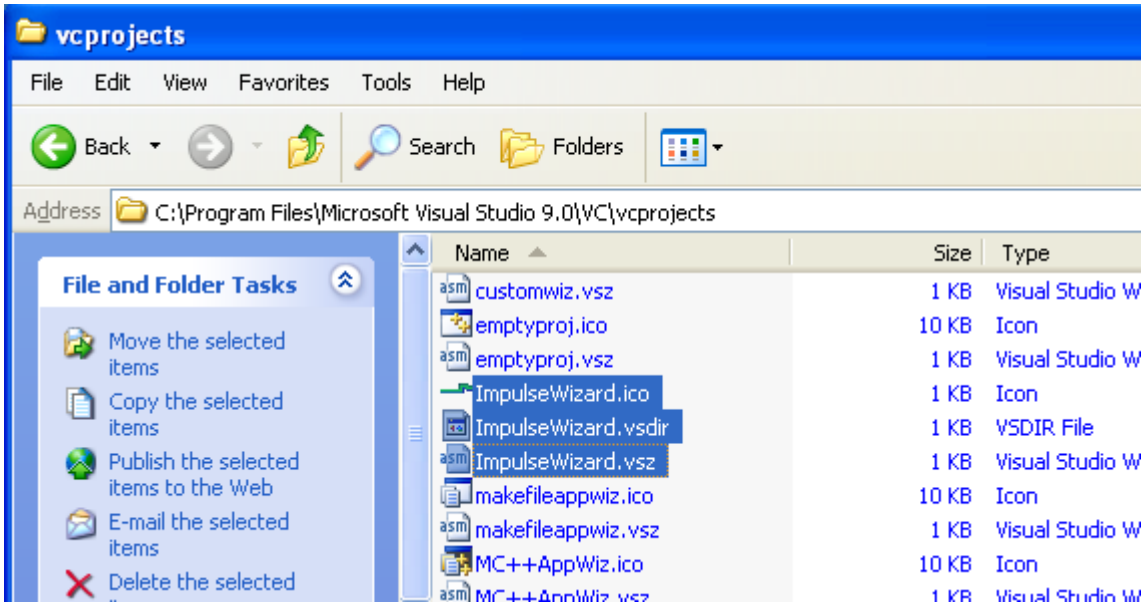


Figure 1, Installation of Plugin

However, if you wish to install it manually install you will need to copy the three files ImpulseWizard and copy them into C:\Program Files\Microsoft Visual Studio 9.0\VC\vcprojects directory.



Step 2: Launch Microsoft's Visual Studio 2008

Launch the Visual Studio 2008 application and close any projects that are already open.

Step 3: Create a New Impulse project within Visual Studio

From the top pull down menu select "File->New Project…"

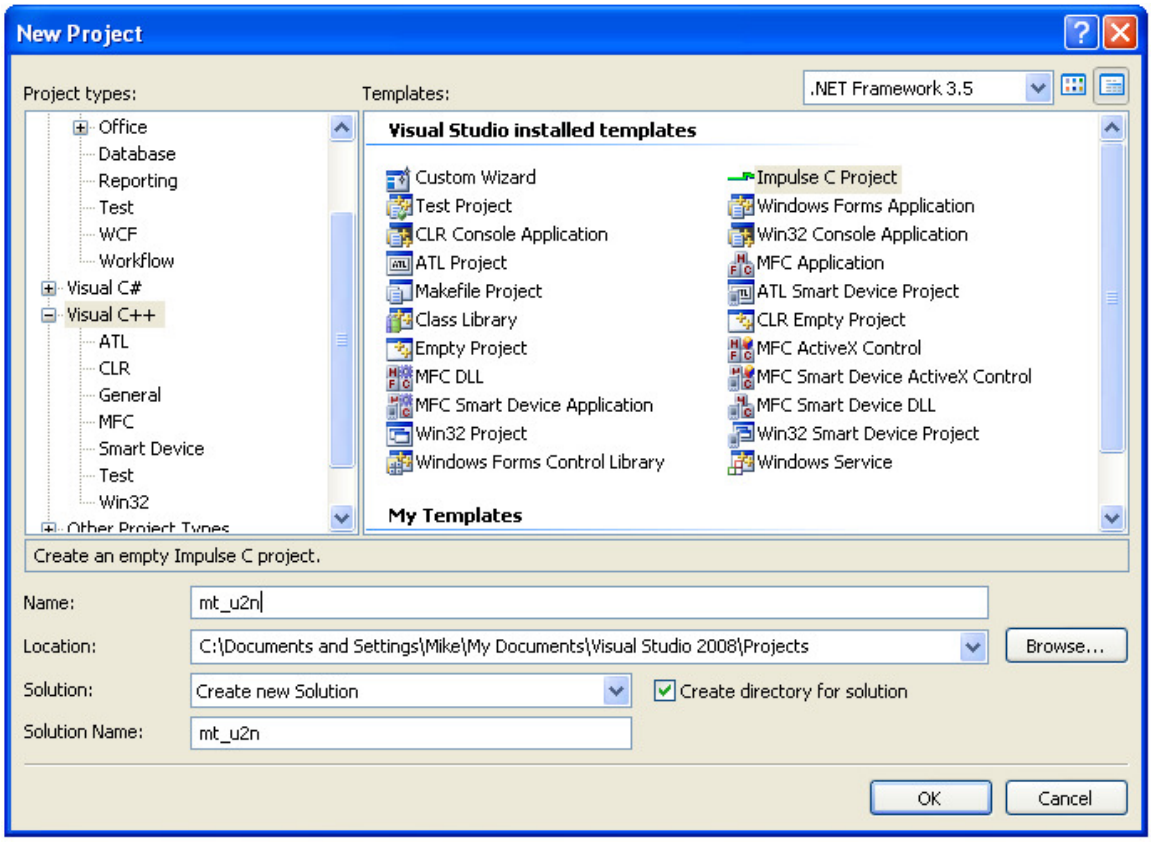


Figure 2, Create a New Impulse Project

Note that when "Create directory for solution" is checked the project will be mt_u2n and the solution will be a subdirectory under the project.

Step 4: Copy source files to the solution directory

For this project we will copy the following files from the mt_u2n (or other Impulse C example) example to the Project directory located in:
"C:\Documents and Settings\Mike\My Documents\Visual Studio 2008\Projects\mt_u2n\mt_u2n"

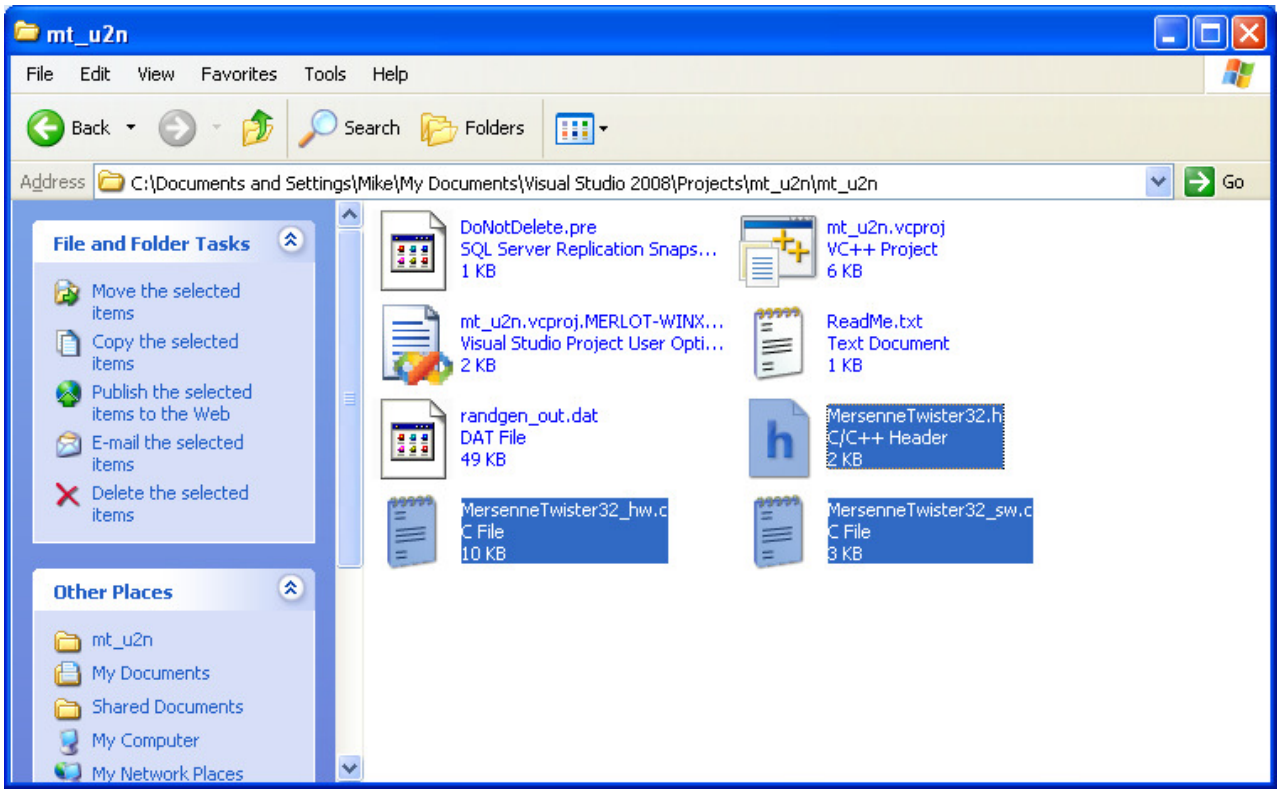


Figure 3, Copy Source files to Project

Step 5: Add header and source files to the Visual Studio Project

Add the header files to the project by right clicking on the Header Files folder in the Solution Explorer – then “Add” and “Existing Item…”

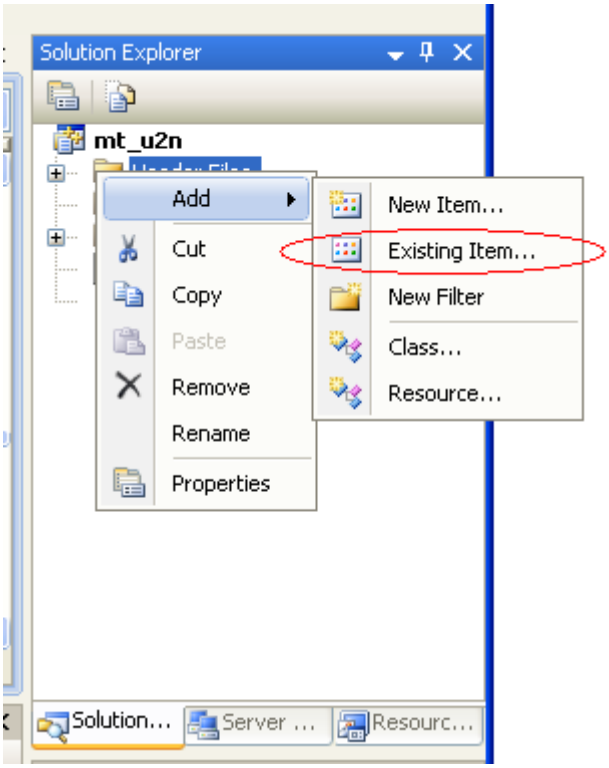


Figure 4, Adding Existing Files to the Project

Next, add the source files to the "Source Files" folder in the Solution Explorer directory. The Solution Explorer will have the following files in its solution.

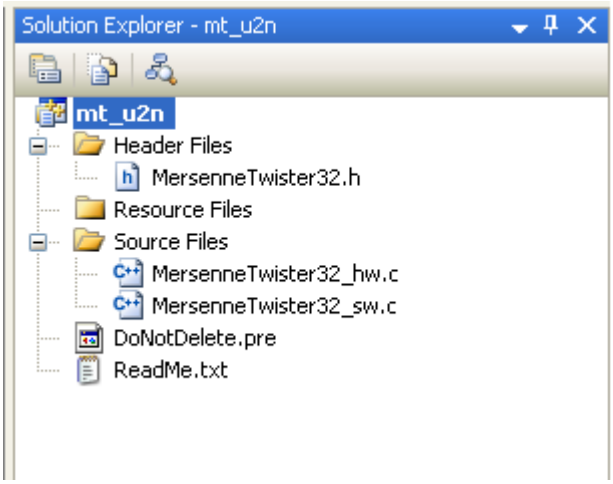


Figure 5, Solution Explorer Source Files

Step 6: Build the "Debug" software project

Building the debug software simulation is important to ensure the application is working correctly prior to hardware generation.

The default Project Configuration is the Debug software build environment. A software simulation can be created and tested by selecting "Build->Build mt_u2n" from the pull down menu.

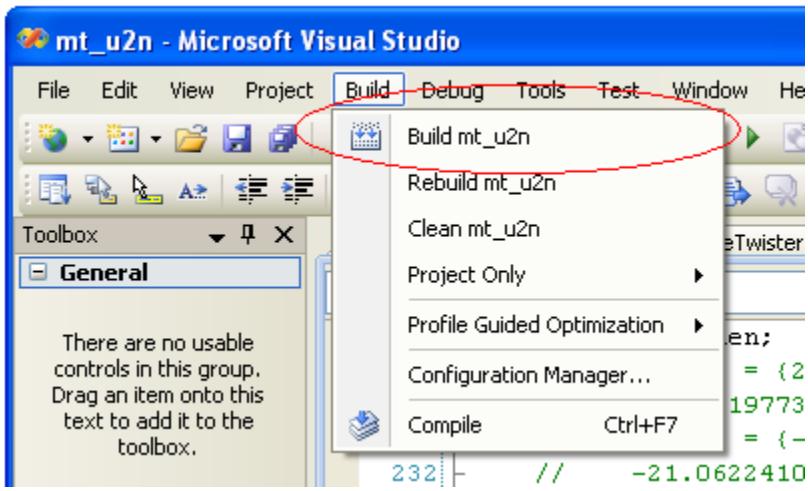


Figure 6, Building the Desktop Simulation

Step 7: Test the application using desktop simulation

The simulation can be run simply by selecting from the top menu "Debug->Start Debugging"

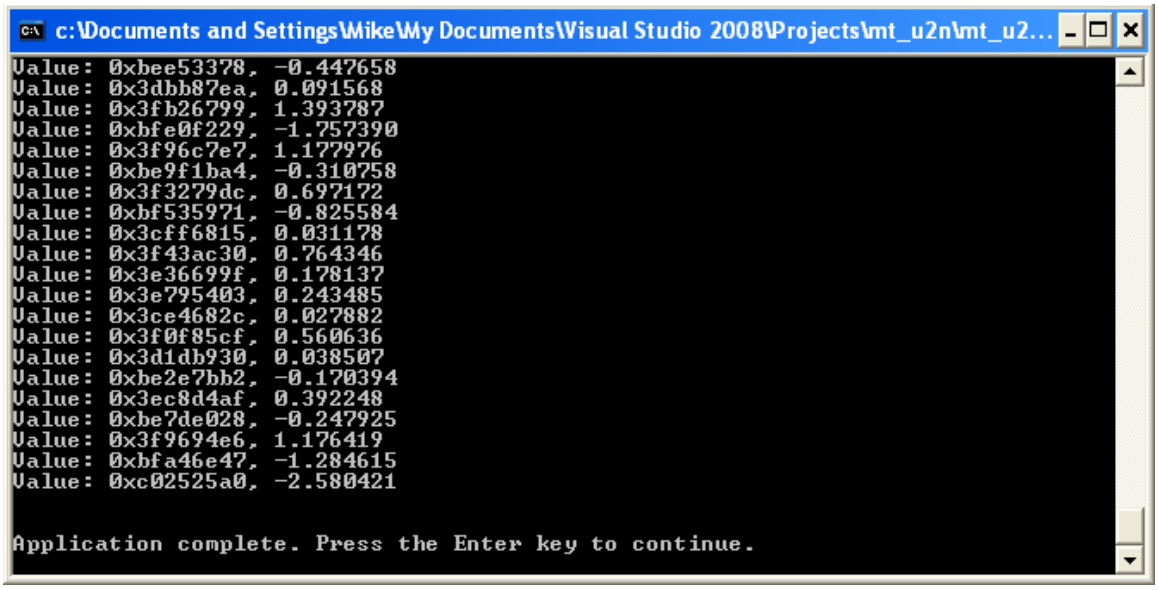


Figure 7, Normal Desktop Simulation Results

The software simulation is working if you get the output shown in Figure 7.

Step 8: Build HDL for the target FPGA

After the project is debugged and working as a software simulation, it's time to generate the HDL.

Select the Hardware environment from the Configuration Manager. To do this, invoke the Configuration Manager Dialog by selecting *Build->Configuration Manager*

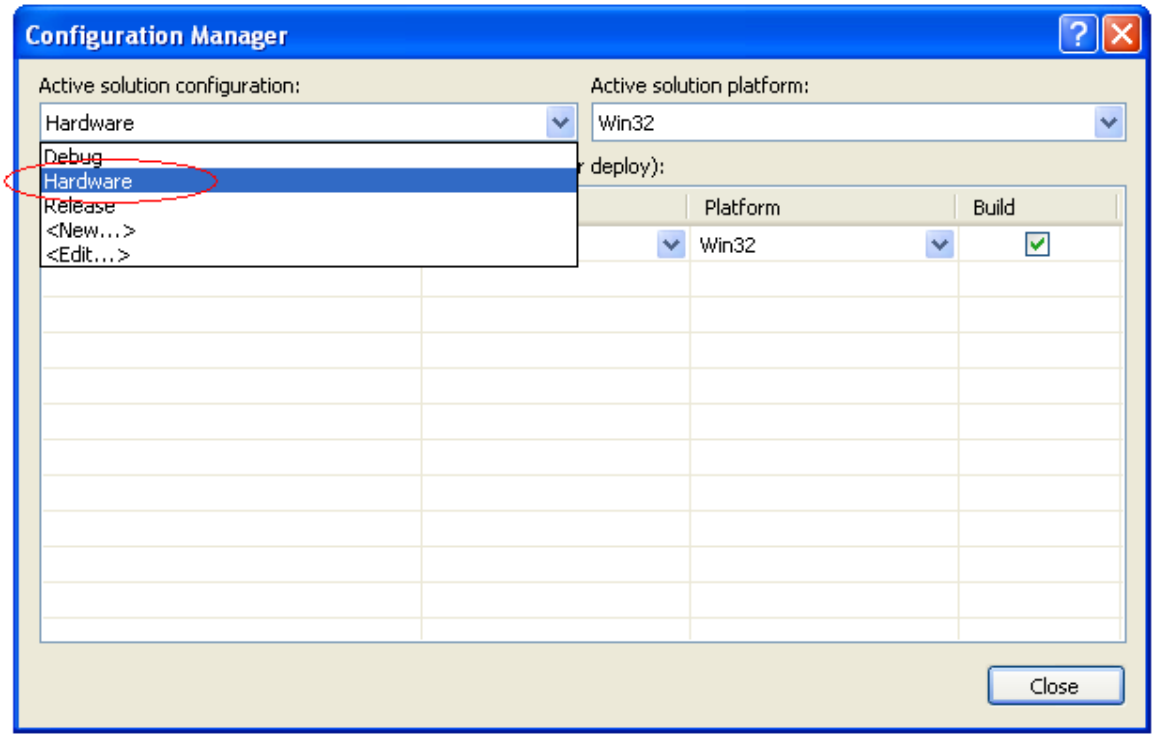


Figure 8, Select Hardware in the Configuration Manager

Step 9: Modify the project properties and add hardware and software files.

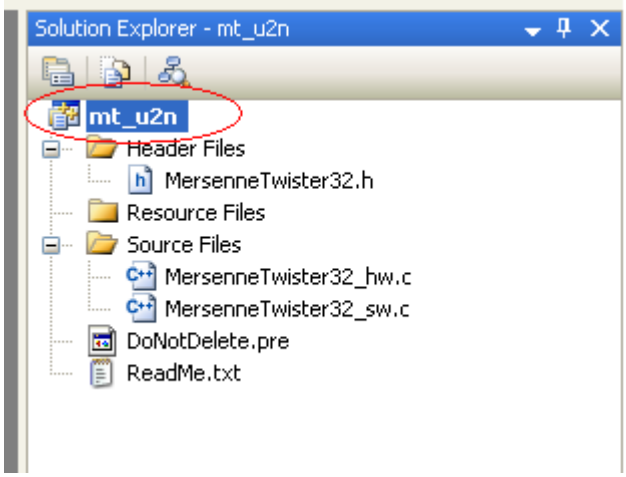


Figure 9, Modifying the Software Simulation Model

a) Select the project by right clicking on the main project in the Solution Explorer.
b) From the menu select the mt_u2n Property Page.
c) Select *"Configuration Properties->Impulse C->Build"* and type in the "*Hardware"* file and the Software files.
d) Follow the table below to verify the proper settings.
e) Click OK when you are finished.

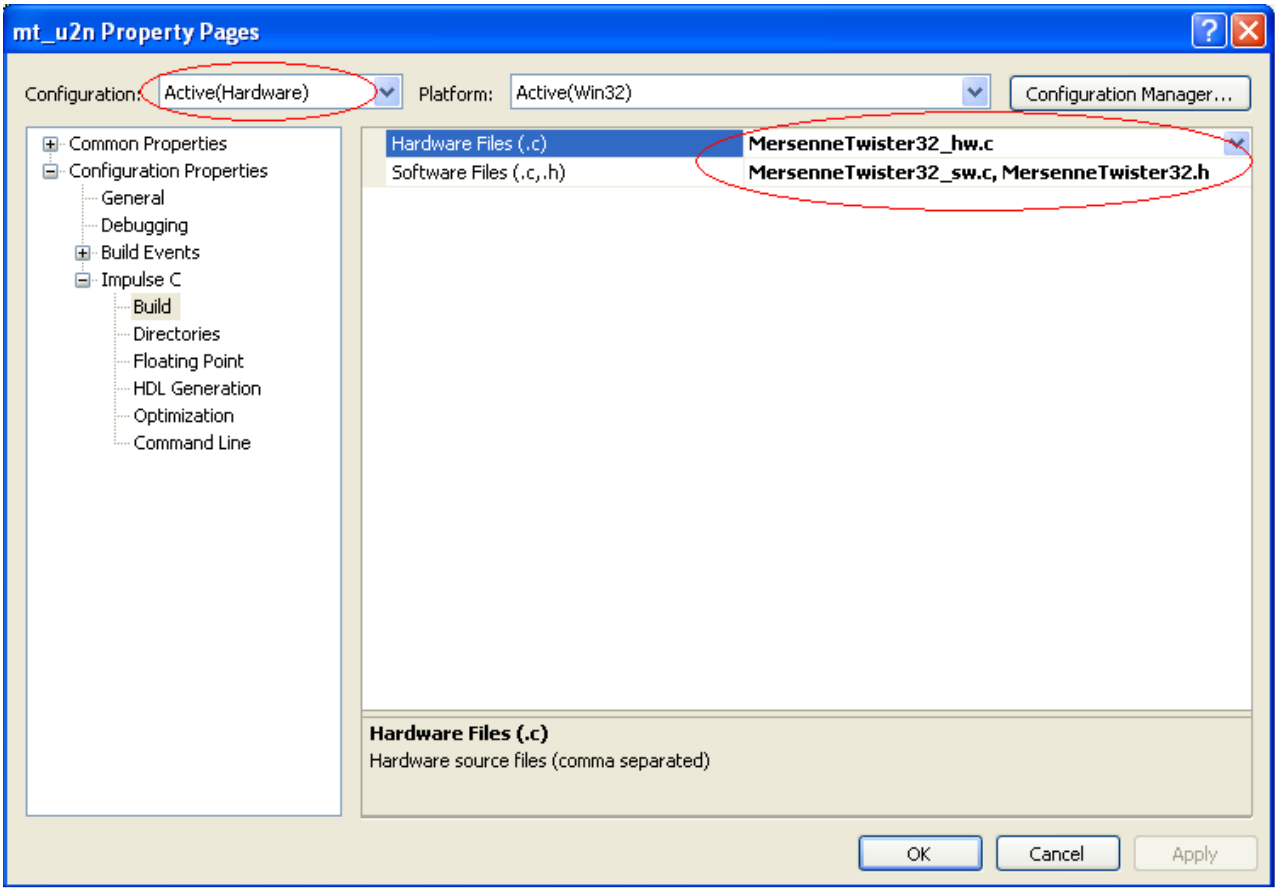


Figure 10, Solution Properties Page

Under "Configuration Properties" there is an "Impulse C" tree element. Listed below is a table showing the required configuration settings.

Node	Key	Value
Build	Hardware Files (.c)	MersenneTwister32_hw.c
	Software Files (.c,.h)	MersenneTwister32.h,MersenneTwister32_sw.c
Directories	Hardware Build Directory	hw
	Software Build Directory	sw
	Hardware Export Directory	export_hw
	Software Export Directory	export_sw
Floating Point	Floating Point Library	None
	Enable Double Precision	no
HDL Generation	Target	Xilinx Generic VHDL
	Dual Clock	No
	Active Low Reset	No
	Use std_logic	No
	Do not include co_ports	Yes
	Export Files	No
Optimization	Constant Propagation	Yes
	Array Scalarization	Yes
	Relocate Loop Invariants	Yes

Table 1, Solution Properties Page

Step 10: Build the HDL hardware

To generate HDL hardware from an impulse project, select “Build mt_u2n…” from the Build pull down menu item. CoDeveloper will launch, examine the project files and build HDL code that can be used to generate hardware for an FPGA.

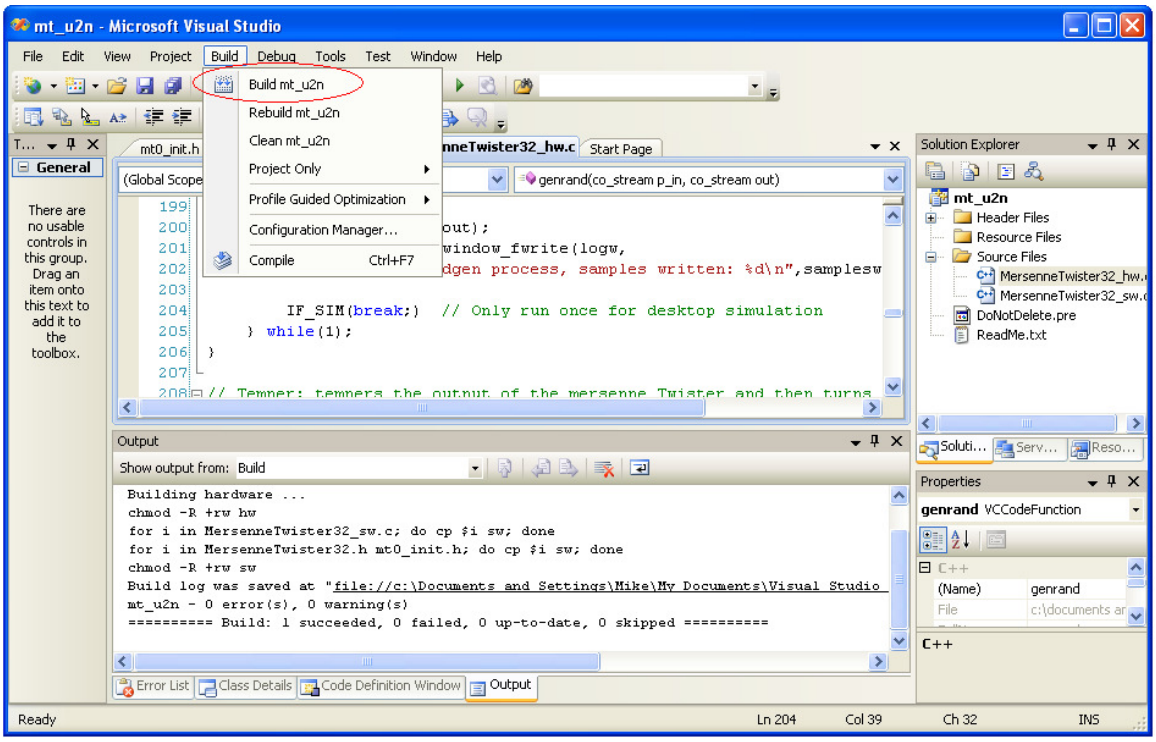


Figure 11, Building HDL

Conclusion

In this application note we have seen how the Visual Studio environment can be used to verify the correct behavior of an application, as well as to generate the HDL files, ready for FPGA synthesis.