Implementing a simple processor-based design in an FPGA

Summary

Tutorial
TU0118 (v1.0) January 29, 2004

This tutorial describes how to implement a processor-based design in an FPGA. It describes the creation of FPGA and Embedded projects, creating C files, setting up processor and compiler options and then configuring and programming the design to an FPGA device.

This tutorial will create a schematic and an embedded project containing a C source file that will be downloaded to the supplied Altera Cyclone EP1C12Q240CB chip on the NanoBoard daughterboard. The resulting bit counter will be displayed on the NanoBoard's LEDs.

Connect the NanoBoard to the parallel port of the PC and power up the board by flicking the ON switch. Make sure you have installed the Quartus tools (web edition) from Altera. These tools can be downloaded from the Altera website (www.Altera.com).

Creating the FPGA project

First we will create an FPGA project.

1. Create a new FPGA project by selecting **File » New » FPGA Project** (or click on **Workspace** button and select **Add New Project » FPGA project**).
2. Save this project by selecting **File » Save Project**, or right-click on the new project name in the Projects panel and select **Save Project As** in a new directory called `Simple_Processor.PrjFpg`.
3. Add a new schematic document to the FPGA project: **File » New » Schematic** (or click on the **Project** button and select **Add New to Project » Schematic**). Save the schematic.

Warning: Do not use spaces or dashes (-) in file names or project names. Use underscores if necessary.

4. Place the following components (listed below in Table 1) on the schematic sheet as shown in Figure 1. Select **Place » Part**, or click on the **Place Part** button on the Wiring toolbar. Each of these components are available from the default generic FPGA integrated libraries, so there is no need to add any libraries for this tutorial. For more information about using libraries, refer to the *Building an Integrated Library* tutorial.

Table 1. Components used in the Johnson Counter schematic

Component name	Available from FPGA integrated library
TSK51A_D (the processor)	FPGA Processors.IntLib
RAMS_8x1K	FPGA Memories.IntLib

Component name	Available from FPGA integrated library
CLOCK_BOARD	FPGA NanoBoard Port-Plugin.IntLib
TEST_BUTTON	FPGA NanoBoard Port-Plugin.IntLib
LED	FPGA NanoBoard Port-Plugin.IntLib
NEXUS_JTAG_CONNECTOR	FPGA NanoBoard Port-Plugin.IntLib
NEXUS_JTAG_PORT	FPGA Generic.IntLib
OR2N1S	FPGA Generic.IntLib
FPGA_STARTUPx	FPGA Peripherals.IntLib

Type the component name, e.g `TSK51A_D`, into the Lib Ref field in the *Place Part* dialog, press **OK** and place the symbol by clicking on the sheet. Right-click to choose another part to place. Stop placing parts by seelcting **Cancel** when the *Place Part* dialog is open.

While the component is still on the cursor and not yet placed:

- press **X** to mirror the component
- press **SPACEBAR** to rotate a component by increments of 90 degrees.

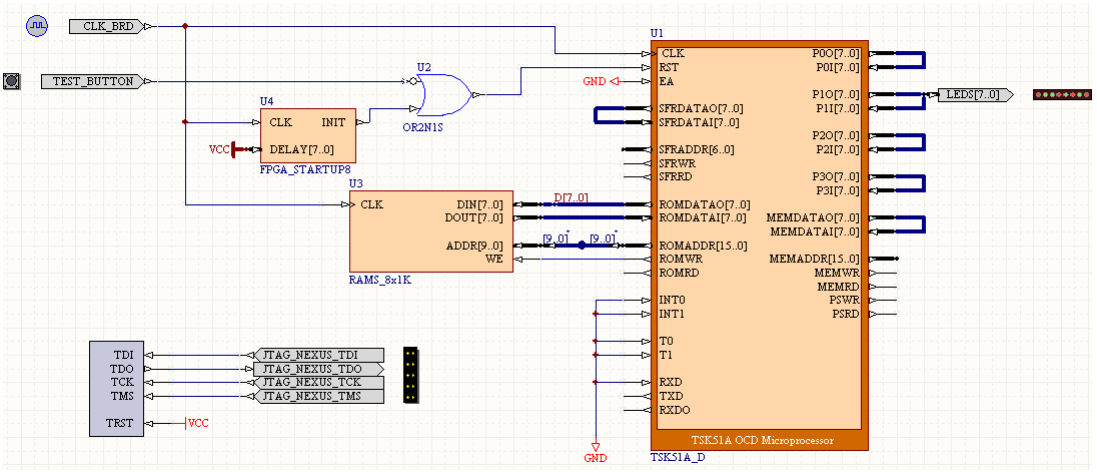
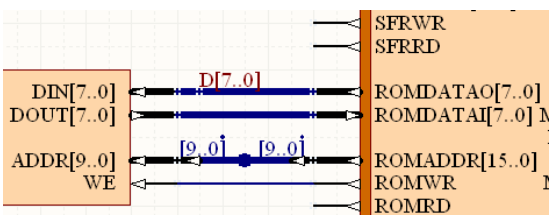


Figure 1. Schematic sheet for an FPGA design

5. Make the connections in the schematic design by using the **Place » Wire** and **Place » Bus** commands.

For more information about placing wires and buses, refer to the *Getting Started with PCB Design* tutorial.



6. Connect the RAMS_8x1K's ADDR[9..0] to ROMADDR[15..0] on the TSK51A-D symbol using a bus joiner. Place the symbol `JB` from

the `FPGA Generic.IntLib` to join the buses, rotating it as you place it using the **Spacebar** key. Modify the parameter text to read [0..9] on both sides of the joiner by using inline text editing mode. Move the parameter text above the bus joiner. The dots above the text indicates that they have been manually positioned.

7. Place net labels on appropriate nets using the **Place » Net Label** command. Press **TAB** to enter the net label name in the *Net Label* dialog before clicking to place the label on the corresponding wire (the cursor changes to a red cross when the net label is correctly attached to a net).
8. Place a GND port on the EA pin of TSK51A_D processor. Select **Place » Power Port** and press **TAB** to enter the power port name (GND) and choose a style (e.g. Arrow) in the *Power Port* dialog before clicking to place. Connect pins INT0, INT1, T0, T1 and RXD to a GND power port as well.

> Use the SPACEBAR to rotate a power port while placing it.

9. Place a VCC power port on TRST of the NEXUS_JTAG_PORT using the Bar style. Click on the VCC icon in the Wiring toolbar. Place a VCC Bus power port on the DELAY[7..0] pin of the FPGA_STATRTUP8 component (its net properties should read `VCCBUS[..]` in the *Power Port* dialog).
10. Annotate the design using the **Tools » Annotate Quiet** command. The *Confirm Designator Changes* dialog displays the following message: “There are 3 designators requiring update. Proceed with changes?”. Click **Yes**. The designators will be changed from U? to U1, for example, automatically and numerically augment.
11. Save `Sheet1.SchDoc` [shortcut **Ctrl+S**] making sure it is in the same directory as the project.

Creating the Embedded Project

Now we can create an embedded project that will hold the source file for the software we want to download to the FPGA chip.

1. Create a new Embedded project by selecting **File » New » Embedded Project** (or click on **Workspace** button and select **Add New Project » Embedded Project**).
2. Save this project by selecting **File » Save Project**, or right-click on the new project name in the Projects panel and select **Save Project**. Use a name without any spaces (e.g. `Embedded_Project1.PrjEmb`) and save it in the same directory as the FPGA project and schematic files.
3. Create a new C file by right-clicking on the embedded project name in the Projects panel and selecting **Add New to Project » C File**, or click on the **Project** button and select **Add New to Project » C file**.
4. Type the following C code into the new C file.

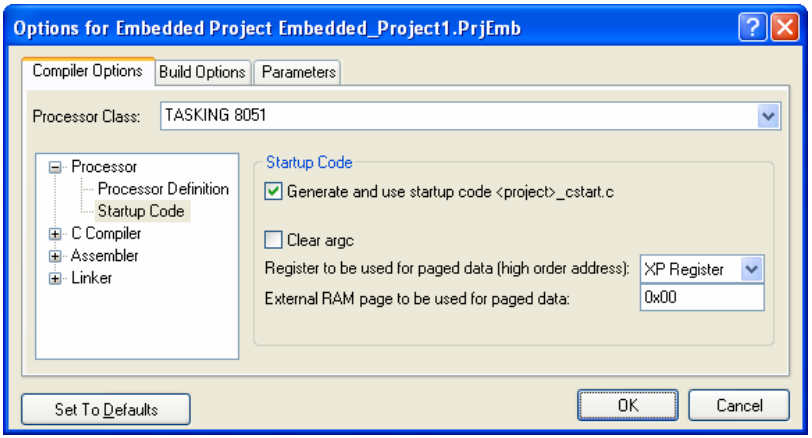
```
void main( void )
{
        unsigned char x = 0;
        unsigned short i;
        for (;;)
        {
                P1 = x++;
                for ( i = 0; i < 0xFFFF; i++ )
                {
                        __asm( "nop" );
                }
        }
}
```

5. Save the code in the same directory as the other project files. Using the default name (`Source1.C`) is fine for this tutorial.

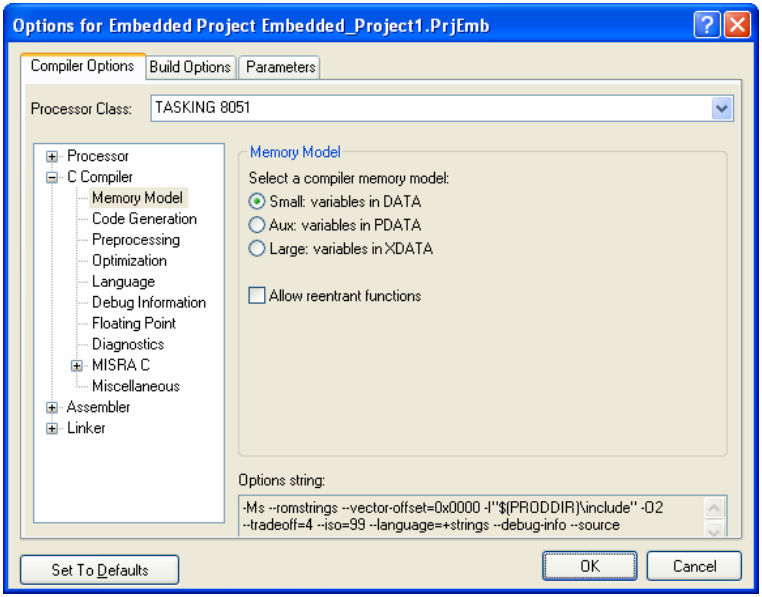
Setting the embedded software project options

Next we will set the processor startup code and set some C compiler options, such as the memory model and code generation options.

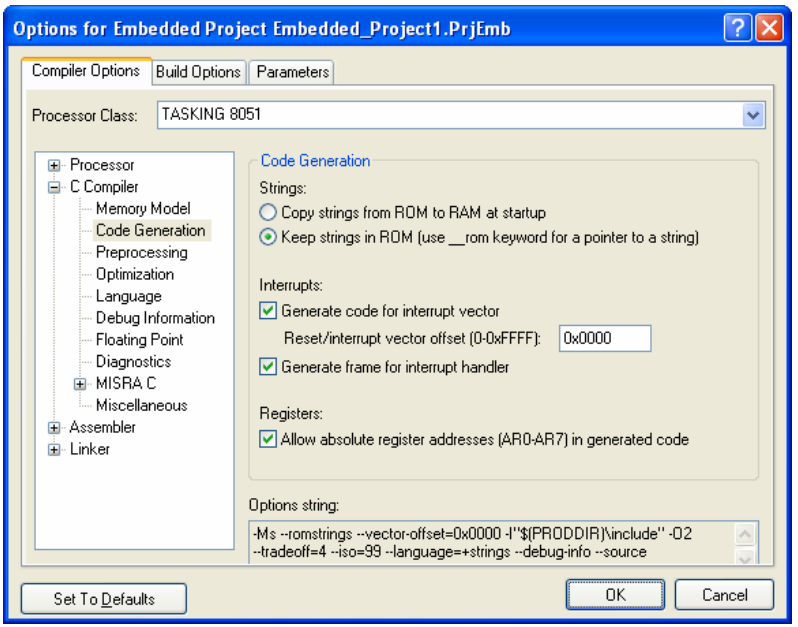
1. Right-click on the embedded project name in the Projects panel and select **Project Options**. The *Options for Project Embedded* dialog opens.
2. Add the startup code to your project by double-clicking on **Processor** and click on **Startup Code**. Make sure **Generate and use startup code <project>_cstart.c** is selected.



3. Set a memory model by double-clicking on **C Compiler**, selecting **Memory Model** and choosing **Small: variables in DATA**. Make sure **Allow reentrant functions** is not selected.



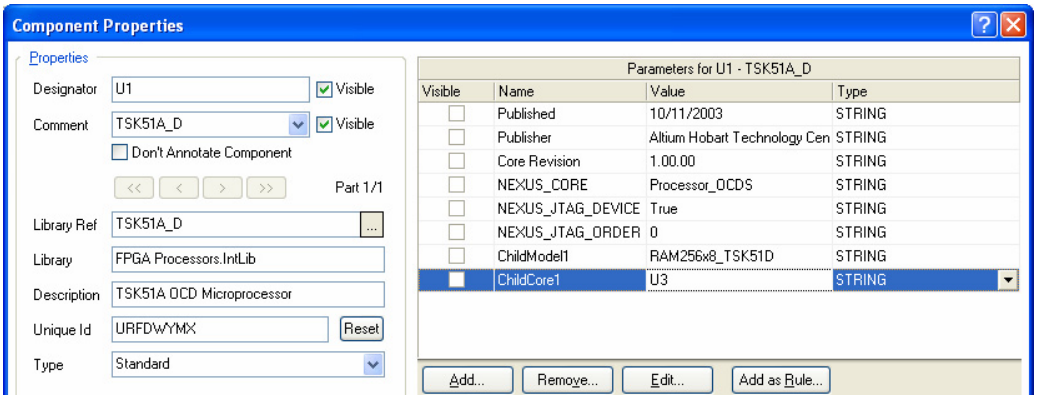
4. While still in the C Compiler options, click on **Code Generation** and set the Strings to **Keep strings in ROM**. Make sure the Interrupts and Registers options are selected. Click **OK**.



Setting the processor properties

Now we need to tell the processor on the schematic where to find its RAM and which sub-design (embedded project) to use when the software is to be downloaded to the FPGA chip on the daughterboard.

1. Go back to `Sheet1.SchDoc` by clicking on its document tab.
2. Double-click on the TSK51A_D processor (`U1`) to open its *Component Properties* dialog.
3. In the Parameters section, enter the designator of the RAM (most likely `U3` — refer to your schematic) in the **Value** field of the **ChildCore1** parameter.



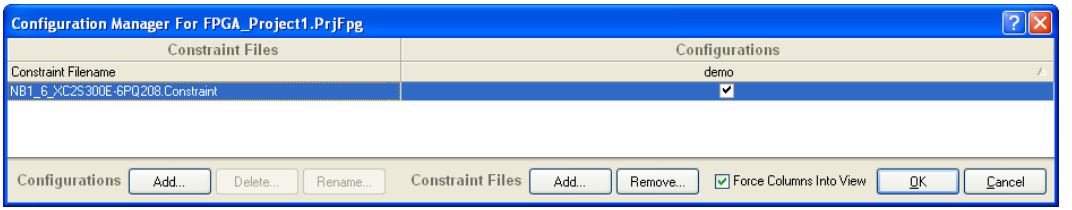
4. To link the Embedded project to the processor component, go to the **Projects** panel and click on **Structure Editor** button to display the project structure view. Make sure that your hardware project is compiled so that it appears in the structure tree.You will see the Valid Sub-Projects list in the bottom section of the Projects panel. Select the software project which we just created from the Valid Sub-Projects list, e.g. `Embedded_Project1.PrjEmb`, by clicking on its icon and drag-and-drop it onto the processor component (TSK51A_D) icon in the top section of the panel. Note that valid targets will be highlighted. The link will be established and the structure will recompile to re-establish the integrity.
5. Save the schematic and FPGA project.

Configuring the design to an FPGA device

Now we need to specify which FPGA chip we want to use in our design, e.g. the Altera Cyclone EP1C12Q240C6 chip on the NanoBoard daughterboard. We will add a configuration and constraint file to do this. The Constraint file will specify the device and determine the pin numbering.

1. To create a new Configuration, right-click on the FPGA Project name in the Projects panel and select **Configuration Manager**, or select **Project » Configuration Manager** from the menus.
2. The *Configuration Manager for project* dialog appears. Click on the **Add** button in the Configurations section of the dialog and type a Configuration name in the *New Configuration* Name dialog, e.g. `demo`, and click **OK**.

3. Add a Constraints file to your configuration by clicking on the **Add** button in the Constraints section and select `NB1_6_EP1C12Q240.Constraint` in the *Choose Constraint files to add to Project* dialog. Constraint files are found in the `Altium2004\Library\FPGA` folder. Click **Open**.
4. Select the **Configuration** checkbox back in the *Configuration Manager* dialog and click **OK**.



5. A folder named `Settings` is added to the project and shows the Constraint file used in the `Constraints Files` folder.
6. Save the FPGA project file.

Using the Devices view to program the FPGA

The Devices view (**View » Devices**) allows you to follow through the workflow (from left to right) required to send your program to the FPGA. In this view, you can:

- Compile the project (and check for errors)
- Synthesize (create an EDIF netlist)
- Build (translate the EDIF files, map the design to the FPGA, Place and Route the FPGA, run a Timing Analysis and then Make the Bit File)
- Program FPGA (download the bit file to the daughterboard's FPGA chip, e.g. the Altera Cyclone).

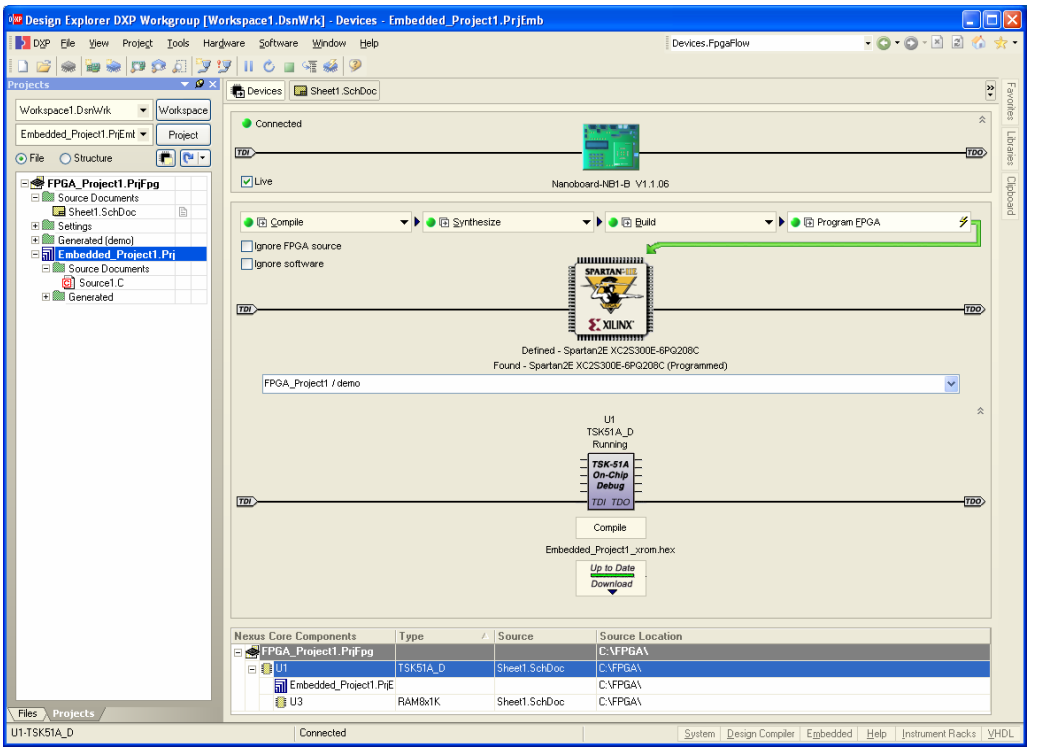
When this workflow is completed, you will be able to run the program by flicking on the DIP switches on the NanoBoard. To download your design to the FPGA:

1. Open the Devices view by selecting **View » Devices**.
2. Make sure your NanoBoard is properly connected and switched on. In the Devices view, click on the **Live** button and check that the **Connected** indicator is green.
3. In the Devices view, click on **Compile**. The red indicator will turn green when a successful compilation takes place. If any error messages display in the **Messages** panel, go back to your schematics and embedded source code, correct any errors, save the files and recompile.
4. Click on **Synthesize**. A *Synthesizing* update dialog will appear. If the synthesis is completed successfully, a folder called `Generated [config_name]` is created which holds the generated EDIF, VHDL and synthesis log file. If the synthesis does not complete, check the **Messages** panel for errors. The configuration which is used in this example, which we named `demo`, will display in the Devices view underneath the Altera Cyclone icon.
5. Click on **Build**. This will step through several processes to ultimately make the Bit file that can be downloaded to the FPGA. You will see the buttons next to the various processes turn green as they are successfully completed.

You can run all stages of the workflow up to and including the current stage by clicking on the arrow icon located on the left side of the stage button, e.g. clicking on this icon on the Program FPGA button will run all previous stages first.

The Build button will turn green when all necessary processes are completed.

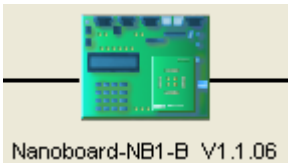
6. Click on **Program FPGA** to download the bit file to the daughterboard's Cyclone chip. Watch the process flow and finally the programming of the FPGA through the JTAG bus.
7. When the Program FPGA process is completed, the LEDs will be flashing as a bit counter.



Setting clock frequencies

The speed (clock frequency) can be tuned on the NanoBoard to slow down or quicken the counter. Let's slow down the counter display.

1. Double-click on the NanoBoard icon in the Devices view.



The *Instrument Rack – NanoBoard Controllers* appears. The default clock frequency is 50MHz.

2. Select another clock frequency, e.g. the slowest (6 MHz), by clicking on the **6(Min)** button.
3. The shifting of the LED display will be slowed down accordingly.

Revision History

Date	Version No.	Revision
29-Jan-2004	1.0	New product release