B5e-Super-Value-Pack Tutorial

This tutorial takes you through setting up your B5e-Super-Value-Pack hardware, and demonstrates some simple projects.

1. Before you start this tutorial.

Before you start this tutorial, I recommend going through the B5-Spartan2e+ Quickstart Guide, if you haven't done so already. You don't need any of the plug-on modules for the Quickstart Guide exercise. Going through the Quickstart Guide first will make sure that your B5-Spartan2e+ FPGA main board is functioning correctly, and that you have your WebPACK software installed correctly.

2. Set up your B5e-Super-Value-Pack hardware.

This is the fun part of unpacking your plug-on modules, and plugging them onto your B5-Spartan2e+ FPGA main board.

Unpack each of the modules, and plug them onto the headers around the FPGA. Each 20-pin header is called a "slot", and there are eight slots labeled A, B, C, D, E, F and G. For this tutorial, use the following slots for the plug-on modules:

Slot	Plug-On Module
A,B	SRAM
C	Switches
D	LEDS
E,F	IDE-Interface
G	Peripheral-Connectors
H	Parallel-Port-Interface

Note that the Parallel-Port-Interface module has long legs, which act as a kind of bridge over the download-pod cable. For this tutorial, the download-pod cable is connected to the JTAG mode 10-pin header, which is next to slot H on the board.

3. Downloading project bitfiles to the FPGA for demonstration.

This section outlines how to download the project bitfiles, which have already been compiled by the Xilinx tools. This allows easy demonstration or testing of the hardware modules, and can also be used to view the expected results, before going through the tutorial steps. All of these project files can be found on the BurchED website.

Download all of the .bit files, and .VHD files (you will use these later in the tutorial), from the BurchED website, to a folder on your machine.

Ensure that header M1 on the CONFIG MODE SEL header block, located in the corner of the B5-Spartan2e+ board near slots B and C, is shorted with a header shunt. This sets the FPGA configuration mode to JTAG.

Connect a regulated +5V DC power supply to your B5-Spartan2e+ board.

Ensure that the red LED on the download pod board, labelled "LOCKED", is off. The toggle switch on the download pod board controls the lock / unlock function. When the system is unlocked, the FPGA can be configured through the cable.

The name of the Xilinx sofware tool used to download the project bitfiles is iMPACT. Double click on the Device Programming icon on your desktop. This starts the iMPACT device programming tool.

A diaglog box will pop up.
Click on:
Configure Devices
Next
Boundary Scan Mode
Next
Automatically connect to cable and identify boundary-scan chain
Finish

The "Assign New Configuration" dialog box will pop up. Navigate through your folders to find the "ChipHeartbeat.bit" file, and double click on it.

A dialog box will pop up with the message "A BIT file describing an XC2S300e is about to be assigned to a device previously identified as an XCV300e. Are you sure you want to do this?"
Click Yes.

The iMPACT application will now show an FPGA icon in its top window, and there will be a message window in the lower half of the screen. The Boundary Scan tab at the top of the screen will be selected, and the FPGA icon will have labels "XC2S300e" and "ChipHeartbeat.bit".

Right click on the FPGA icon, and select Program…
A dialog box will pop up. Make sure that the Verify checkbox is unticked.
Click OK.

A progress bar will now pop up, indicating that the bitfile is being downloaded to the FPGA. Download times may vary, but in general it will take about 17 seconds for the configuration to complete. A Programming Succeeded message will pop up, when the download is complete.

You will now notice that the red LED on your B5-Spartan2e+ board is flashing at about 1Hz! If you press the test pushbutton switch on your board, the LED will stay on. When you release the pushbutton, the LED will flash again.

Do:
File
Save As…

Navigate to the folder where you have downloaded your bitfiles
Enter File name = ChipHeartbeat
Click on Save

This will save the iMPACT .cdf settings file to your folder, so that you can easily bring up this configuration later, and redownload the ChipHeartbeat bitfile.

Do:
File
Exit

You can now go ahead and download the other project bitfiles, using the same procedure, to demonstrate the other projects.

The next section of this tutorial will take you through the process of creating and compiling the source files for these projects.

4. Introduction to components.

VHDL and Verilog design is normally done by partitioning the total design into functional blocks called components. Components are simply VHDL or Verilog code modules that can be instantiated in a design, and connected to other elements or components, to make up the total design.

In this tutorial, we will run through a VHDL example of how to create a component, and then we will use this component in a larger design at the "chip level".

We will then go on to use multiple components, supplied for this tutorial, in constructing the example applications.

Once you can see how this is done, you will be able to construct you own components, and use other available components, to build designs. Sometimes collections of components are called "component libraries", but in strict VHDL syntax, a "library" is a collection of precompiled design units.

5. How to design components.

Designing a component involves creating a new project in WebPACK, writing the code for the required function, and performing some verification and checking (in this tutorial, we will just be synthesising the code, to verify the code syntax and tool flow).

It is recommended, where possible, to write the component using generic VHDL or Verilog, without instantiating device specific elements, such as pads, buffers, delay-locked-loops etc. Of course, in some cases this is necessary, to implement the required function. However, in most cases you can write the code generically, which will increase portability and reusability across devices and projects. If necessary, device specific elements will normally be specified at the top/chip level of the code, where all of the cores are instantiated and connected together. In this tutorial, we will be calling the top level "Chip…", since this is the whole chip level.

Note also that, when writing your code, you should include references in the header section, which give credit to any sources that you have referred to when developing your design and writing your code. Cite any textbooks, articles, code, web pages or other resources that you have used. Always "give credit, where credit is due".

6. Compiling the Heartbeat component.

Start:
WebPACK Project Navigator

Type in a new Project name:
Heartbeat

Select a folder for your project eg.
c:\Projects\Heartbeat
(you will need to create this new folder with Windows Explorer ie. outside of the WebPACK environment).

Device Family = Spartan2E
Device = 2s300e-6pq208
Design Flow = XST VHDL
OK

At this stage, if you were going to create your own new component, you would then do: Project, New Source…, VHDL Module, File Name = YourComponentName.

However, for this tutorial, we are going to just add the Heartbeat.VHD source file to the project.

Do:
Project
Add Source…

Then navigate to the folder where you downloaded the .VHD files (in part 2 of this tutorial), and select Heartbeat.VHD
Choose Source Type = VHDL Module
Click OK.

Double click on "heartbeat" in the Sources in Project window. This will pop up a window, showing the Heartbeat.vhd source code.

Scroll through and have a detailed look at the source code for this component. When writing your own components, you can use this, or other, existing component source code as a template to get you started. This saves some time, especially in formatting the header of the file.

The next step is to compile this code, which allows us to debug and check the syntax of the code, and also to determine the device resource usage for this component. Device resource usage is measured in "slices". A slice, in Xilinx terminology, is equivalent to a four-input lookup table, plus some carry logic, plus a flip-flop. This number gives us some kind of basic handle on how big the component is, after it has been synthesised and compiled.

Note that we are not assigning any pinouts of the device at this stage, because we are only working at the component level. We will assign pinouts at the Chip level later.

Double click on the Implement Design process, in the Processes For Current Source window.

The compiler will run, and you will get a green-tick on the Synthesis process, and a yellow-exclamation-mark on the Implement Design process. The yellow-exclamation-mark is OK. It just means that a warning has been generated in a process. Warnings are not necessarily design errors. You should always view the warning messages in the report files.

Click on the "+" next to the Implement Design process to expand out the sub-processes (Translate, Map, Place & Route).

Click on the "+" next to Place & Route, to further expand.

Double click on Place & Route Report. Browse through the report file. Note in the Device Utilization Summary section that this component uses 23 slices.

7. Creating the chip level code, and instantiating components.

We are now going to create a new project, where we will instantiate the Heartbeat component, and also assign pinouts for the device. Then we will compile the full "Chip" design, and download it to the FPGA.

Do:
File
New Project…
Type in a new Project name:
ChipHeartbeat

Device Family = Spartan2E
Device = 2s300e-6pq208
Design Flow = XST VHDL
OK

Do:
Project
Add Source…

Then navigate to the folder where you downloaded the .VHD files (in part 2 of this tutorial), and select ChipHeartbeat.VHD
Choose Source Type = VHDL Module
OK

Double click on "chipheartbeat" in the Sources in Project window. This will pop up a new window, showing the ChipHeartbeat.vhd source code.

Scroll through and have a detailed look at the source code. Note that the Heartbeat component is declared at the top of the architecture section of the code (this section is called the "declarative section" of the architecture, where all of the components and signals are declared). Then in the body of the architecture, the Heartbeat component is instantiated, and connected up to the signals at this level.

We now need to also add the Heartbeat.VHD module to this project, so that the compiler can include the actual Heartbeat component VHDL code in the compilation.

Do:
Project
Add Source…

Then navigate to the folder where you downloaded the .VHD files (in part 2 of this tutorial), and select Heartbeat.VHD
Choose Source Type = VHDL Module
OK

Note that the Sources in Project window now shows the two source files, and that ChipHeartbeat contains the Heartbeat file in its heirarchy tree.

8. Assign the pinouts for the chip, and compile the design.

The pinouts, or pin numbers, of the chip are assigned by editing the "user constraints file" (UCF) in the project.

Highlight the three lines, with your mouse, in the ChipHeartbeat.VHD header file that start with "–NET…". Do Ctrl-C, for copy to clipboard.

Go to the "Processes for Current Source" window in the design environment, and expand out the processes, so that you can see the Edit Implementation Constraints File process. Double click on this process.

A notepad window will pop up, with the UCF file. Paste the three lines at the very bottom of the file (using Ctrl-V). Edit these three lines to delete the leading "--" characters at the start of each line. Save the file, and close the Notepad window.

A dialog box with a message about the UCF file will pop up.
Click Reset.

Double click on the Implement Design process, to run the compiler. You will get green-ticks and yellow exclamation marks on the processes as they complete.

Double click on Place & Route Report. Browse through the report file. Note that in the Device Utilization Summary section, this chip design uses 24 slices.

9. Create the programming file.

Right click on the Generate Programming File process, and select Properties…

Click on the Startup Options tab.

Set:
Start-Up Clock = JTAG Clock

Click OK.

Double click on the Create Programming File process. When the process completes, you will get a green-tick.

9. Download the design to the FPGA.

Connect the power supply to your B5-Spartan2e+ board.

Double click on the Configure Device (iMPACT) process. This will start the iMPACT device configuration tool.

Right click on the Xilinx FPGA icon, and select Program…

A dialog box will pop up. Make sure that the Verify checkbox is unticked. Click OK.

A progress bar indicator will pop up, which shows the progress of the download of the FPGA configuration bitstream. When finished, you will get a "Programming Succeeded" message, and the LED on your B5-Spartan2e+ board will be flashing at about 1 Hz. If you press the test pushbutton switch on your board, the LED will stay on. When you release the pushbutton, the LED will flash again.

Do:
File
Exit

At this stage there is no real need to store the iMPACT settings, because they are reloaded when launching iMPACT from the Project Manager design environment. So when the "Save settings…" dialog box pops up, just click No, and the program will close.

You can now go ahead and compile the other "Chip…" VHDL project source files, using the same procedures. Study the VHDL source code to see how multiple components are instantiated and connected together in a single chip design, to create the total design.

This tutorial will be updated intermittently, with extra material, and incorporating feedback from customers. Please check on the website for updated versions. The date at the top of this document indicates when it was last updated.

10. Support.

Please email Tony Burch, tony@BurchED.com.au, with any questions, comments, feedback, or requests for support.

Thanks for purchasing the B5e-Super-Value-Pack. We hope that you will enjoy using the units in learning, creating new ideas, and prototyping your designs. ☺

References:

1. M. Keating and P. Bricaud., Reuse Methodology Manual for System-On-A-Chip Designs, Kluwer Academic Publishers, 2000.

2. J. Hamblen and M. Furman, Rapid Prototyping of Digital Systems, Kluwer Academic Publishers, 2000.

3 D. J. Smith, HDL Chip Design, Doone Publications, 1996.

4. S. Sjohlm and L. Lindh, VHDL for Designers, Prentice Hall Europe, 1997.