DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF BRITISH COLUMBIA
CPEN 391 – Computer Systems Design Studio
2015/2016 Term 2

Tutorial 1.2: Introduction to QSYS

In this tutorial, you will learn how to use the QSYS Design tool from Altera. QSYS is integrated into Quartus II. Before starting this tutorial, you should make sure you are familiar with Quartus II, and can use it to create simple VHDL or Verilog designs, compile the designs, and download the designs to an FPGA on the DE2 board. If you need a refresher, you should review your notes from EECE 353 or equivalent

The attached pages are from Altera, and provide an excellent introduction to the tool. In working through this tutorial, I have made some additional clarification notes, listed below.

Overview:

From previous experience, some people get confused between the various levels of design. Your final design will consist of hardware (a processor and peripherals) and software (written in C) running on that processor. This overview of the design process might help clarify things. Figure 1 shows the overall flow, and is explained in text below:

1. In the tutorial, you first will use the GUI in QSYS to define a system containing a processor and peripherals (Section 4 of the tutorial). At the end of Section 4, you click "Generate" which generates a Verilog description of the system containing the processor and peripherals.

2. You then compile these Verilog files, along with a top-level VHDL or Verilog files, using Quartus II just as you did in EECE 353 (Sections 5 and 6 in the tutorial). This produces a bitstream file that describes the hardware of your system (processor and peripherals). You may also include your own hardware to attach to the processor (more on that in Module 2). You can configure the FPGA on the DE2 board using this hardware using Quartus II (again, as in ECEE 353).

3. You then write C software to run on the processor using a standard text editor. You then use the Altera Monitor Program (AMP) to compile the C code, and to download the C code to the embedded memory on the FPGA. In a later tutorial, we will replace AMP with a better tool.

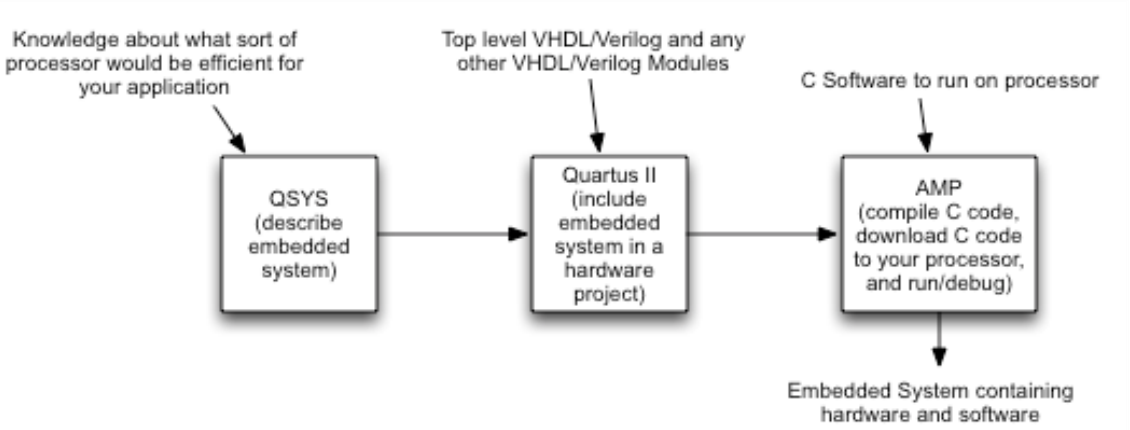

Figure 1: Overall Design Flow of an Embedded System Containing Hardware and Software

Other comments specific to the tutorial sections are listed below. Please read these along with the tutorial.

Section 4 Step 1.

Most of you are using the DE2 board rather than the DE2-115 board, so be sure to choose the appropriate chip from Table 1 when you create a project.

Section 4. Step 6

In Step 6, you connect the basic components to each other by turning on "connection points". For large designs, it can be easy to miss a connection. Be sure you compare your designs to Figures 9, 10, 11 carefully as you work through the tutorial to ensure you have completed the connections properly.

Section 5.1:

Section 5.1.1 shows how to create a top-level design in Verilog, while Section 5.1.2 shows how to create a top-level design in VHDL. You can choose either language for your top-level design (but not both). My suggestion is to use VHDL (since you have used it in EECE 353), so use the file in Figure 23, and ignore the file in Figure 22.

Section 6:

The tutorial asks you to include *nios_system.qip* in your project before compiling. This file can be found in the *nios_system/synthesis* directory (within your project directory). You will need to show all files, since by default, qip files are not shown. Also, remember that you should load in the pin assignments file before compiling (as in EECE 259 and 353).

Section 7:

This section shows how to write software to run on the NIOS processor. You can write software in either Assembly Language (as you might have done in EECE 259) or C. I recommend you use C for all your programming in this course. In that case, you would read Section 7.2 (but not Section 7.1).

In Figure 25, the code assumes that the "switches" port is mapped to location 0x0002000 and the "leds" port is mapped to location 0x0002010 (remember that "0x" means the number is in hexadecimal). Be sure these addresses match those you specified when building the NIOS system in Section 4 (see Figure 17, column "base"). Since, in the tutorial, you are doing an automatic assignment for base addresses, it is possible that QSYS will choose different base addresses. If they do not match, you should modify your C code appropriately.

The programming method you will use is somewhat different than that described in the tutorial. Some IP cores supplied by Altera are not free. However, Altera provides a free "evaluation method" for using these cores. Systems containing these cores can be run as long as the DE2 board is connected to the host using the JTAG cable (the cable you use to download your design). However, such designs must be downloaded from Quartus II (as you did in EECE 353) rather than the Altera Monitor Program (as in the tutorial). To do this:

a) Before running the Altera Monitor Program, download your design from Quartus II just as you did in EECE 353. After you download, a window that says "Click Cancel to stop using OpenCore Plus IP" will pop up. *Do not close this window.* The evaluation version of the cores will only work if this window is open, and the DE2 is connected to the host using the JTAG cable. If you do not see this window, it is because you are only using free cores.
b) In the Altera Monitor Program, leave the box "Quartus II programming (SOF) file (optional)" empty (see Figure 28). At the bottom of Page 32, the tutorial tells you to "When a pop-up box asks you if you want to have your system downloaded onto the DE-series board click Yes.". Since you have already downloaded the file, you should click "No" instead.