CSEE W4840 Embedded System Design Lab 4

Stephen A. Edwards

Due February 26, 2004

Abstract

Forget everything you learned about C and implement the first lab again, this time in pure VHDL. Count in hex on the seven-segment display. Learn to command the Xilinx VHDL logic synthesis tool.

1 Introduction

For labs 1–3, you treated the XSB-300E as mostly a target for C programs. If this were our only objective, we would have chosen a board with, say, a small processor and a few peripherals. Instead, the board has a very flexible FPGA and beginning with the lab, you will get a chance to take advantage of this flexibility by designing and implementing your own hardware.

In this class, you'll be using VHDL (VHSIC Hardware Description Language) to describe hardware. It is a fairly verbose language, but fairly simple at its core. You will want to consult the *Writing VHDL for RTL Synthesis* handout available on the class webpage for examples of how to write VHDL. Unfortunately, the language is very big and large parts of it cannot be directly transated into hardware. As such, things like the language reference manual and the majority of VHDL books are useless because they are too complicated and largely irrelevant for specifying hardware.

Although the syntax of VHDL vaguely resembles that of an imperative language like C, do not be deceived: *VHDL is not a programming language*. In particular, the sort of imperative, algorithmic thinking that works well to solve problems in C *will not* work in VHDL. Specifically, a C-like VHDL program probably will not compile and even if it does, you will not like what you get.

VHDL is closer to a purely structural language. In VHDL, you mostly define how components connect. The main idea is that a system is composed of hierarchically-arranged blocks called entity/architecture pairs (everything in VHDL has a weird name, unfortunately). For each block, you define its interface (a list of wires that enter and leave it) and its guts, which may consist of instances of other blocks, dataflow expressions (e.g., a particular signal is the logical AND of two others), and processes that appear to contain imperative code.

2 The Assignment

As usual, we have provided a skeleton of the code for this lab, located as usual at `~sedwards/4840/lab4.tar.gz` on the ilab machines. This tarball contains a Makefile, two VHDL files (suffix ".vhd"), and a few configuration files that together produce a bitstream for the FPGA that counts and blinks the LEDs on the XSB-300E board. "make download" will compile the VHDL source files, place and route them, produce a bitstream, and download it to the FPGA.

Your assignment is to implement the behavior of the first lab in VHDL, i.e., make the LEDs display a human-speed count from 0–99 in decimal. The hello.vhd file contains a few clock dividers (the default clock on the XSB-300E runs at 50 MHz, slightly faster than your eyes could follow), connections to the LED ports, and two instances of a trivial module called "hex2led," defined in hex2led.vhd.

You need to changed hex2led to decode the seven-segment displays and adapt hello.vhd to emit a decimal count (it currently counts in hexadecimal).

Show your working solution to a TA, have him sign off on it, and turn in a listing of your VHDL source files.

As usual, short, elegant solutions (as much as possible in VHDL) will receive better grades than messy ones.