Methodology

April 19, 2017

0.1 The role of this segment I am adding

This is a modification on the methodology stated out in the report. It specifies the development board we are to use and expounds on the role of the mentioned logic simulators and the synthesis tools.

0.2 Modified methodology

To begin with, we shall use Solaris a Unix Operating System backed by a SPARC workstation from SUN. We considered the use of a Virtual Machine and opted to explore the installation of Linux in a Virtual machine. The Preliminary investigations into doing this project revealed that we had three ways to do this: Parallels Desktop, VMware Fusion and VirtualBox and chose to go for VMware Fusion. With the two major FPGA vendors Altera and Xilinx, we opted for Xilinx; largely due to the MicroBlaze soft processor. The development board of choice is to be between the ML403 board that has a Virtex-4 FPGA with a PowerPC 405 core. The embedded processor to be used will be the Micro Blaze soft processor core: a 32-bit Harvard RISC architecture optimized for vendor-of-choices (Xilinxs) FPGAs. The basic architecture consists of 32 general-purpose registers, an Arithmetic Logic Unit, a shift unit and two levels of interrupt. The FPGA design software packages we will use are bundled on the ML403 board and are called Integrated Software Environment (ISE) and Embedded Design Kit (EDK) and will be used to design and implement the additional software intended for exploring simple processor designs. The logic simulators, which come in handy in verifying whether our designs on the custom FPGA we are to work with, are correct; saw us explore the one provided by Xilinx software which is Verilog and VHDL Simulator but this runs under only Windows XP. So we opted to use Incisive Unified Simulator which uses Verilog-XL, A Candence-owned Hardware Description Language but only after obtaining an evaluation license from Candence. The Synthesis tool to be utilized will be from Xilinx but XST Synthesis tool seems like a sufficient one to use for turning an abstract form of the desired circuit behavior into a design implementation in terms of logic gates. We will also use C programming Language substantially as most of the Xilinx software device drivers are written in C.