Project Delta
An Interactive FPGA Circuit Simulator

Progress Report

Developers:
Robert Duncan – rad55@cam.ac.uk
Justus Matthiesen – jm614@cam.ac.uk
David Weston – djw83@cam.ac.uk
Christopher Wilson – cw397@cam.ac.uk
Rubin Xu – rx201@cam.ac.uk

Client:
Steven Gilham
steven.gilham@citrix.com

February 11, 2009

Contents

1 Project Change

A serious problem with the *JOP* processor's implementation of the standard Java library means that we have had to change our approach to the problem. The processor does not have any built-in support for either *Reflection* or *Serialization*, meaning that since our project was designed on the assumption that these library classes were implemented, the whole structure of the simulation accept has had to have been changed. The decision was made to instead perform most of the simulation on the PC and then use the serial link to communicate LED and switch changes with the DE2 board. This change still leads to effectively the same application, one where one designs a circuit on the PC and then interacts with it using the LEDs and switches present on the board. To enable the user to still be able to program the board and then disconnect it, we are providing an "export to Verilog" feature, that will produce an Altera project file ready for loading onto the board.

2 Current acheivements

We have currently acheived a significant amount, however we are slightly behind schedule in beginning the testing phase of the project.

Every member of the team has contributed well to the project performing tasks assigned to them within reasonable time frames.

2.1 Robert Duncan

(a) Project management and document compilation.

(b) Set up Google Code project repository.

(c) Wrote interface between the simulator and the board.

(d) Mostly completed ComponentGraph to Verilog translation.

(e) Wrote file-handling code to deal with Quartus project for Verilog output.

(f) Created Gate SVG diagrams.

2.2 Justus Matthiesen

(a) Implemented a (non-hierarchical) gate-level circuit data structure based on a directed graph (provided by the JGraphT library).

(b) Implemented a hierarchical logic-component-level data structure for circuits building on top of the gate-level data structure mentioned above.

(c) Created several classes representing boolean function (using an additional state X) and their equivalents in gates (including pseudo-gates for LEDs and switches which rely on the interface to the board written by Robert Duncan).

(*d*) Implemented a discrete time (event-driven) simulator for digital logic circuits. It simulates at the gate level and uses unit-time gate delays.

(*e*) Investigated the use of several GUI libraries for enhancing the GUI (e.g. SVG images).

2.3 David Weston

(*a*) Created classes for the main window, component panel and clock panel.

(*b*) Have used these to produce a window with the necessary panels, toolbars and menus.

(*c*) Created temporary icons.

(*d*) Designed a data structure to use to display components in the panel and a temporary method of displaying them (still need to implement concertina-style menu).

(*e*) Integrated circuit panel into the window, with buttons on the toolbar successfully performing all the actions that have so far been implemented.

2.4 Christopher Wilson

(*a*) Created a panel for the circuit diagram based around an example application that came with JGraph.

(*b*) Implemented Undo and Redo functionality for the diagram.

(*c*) Implemented Cut, Copy and Paste for the diagram.

(*d*) Created Action classes for all of the above so they can easily be added to toolbars and menus.

(*e*) Created "model" classes for circuit components (just AND and OR initially).

(*f*) Created "view" classes for those gates to contain visual information about the gates themselves.

(*g*) Created separate classes for Input and Output ports on components.

(*h*) Created view classes for Input and Output ports and created a renderer to display them as coloured blobs.

(*i*) Note: most of the above classes were created by subclassing what is already in the JGraph library.

2.5 Rubin Xu

(*a*) SDRAM controller for JoP.

(*b*) Memory mapped I/O of LEDs and switches for JoP.

(*c*) Communication link over USB-Blaster for JoP.

(*d*) Win32 low level communication Java class via JNI and corresponding wrapper DLL.

(*e*) High level communication class at PC side and daemon Java program on JoP side to allow controlling of switches and LEDs on DE2.

3 Still to implement

(*a*) Integrating simulator functionality with board.

(*b*) Improve drag and drop functionality, including adding concertina effect on component library.

(*c*) Improving method for creation of wires.

(*d*) Improve look and feel of the GUI.

(*e*) Complete Verilog implementation to reflect components added.

(*f*) Modular testing on the simulator package as well as mocking for the GUI.

(*g*) System testing.

(*h*) Documentation and help files.

4 Testing Strategy

Testing has already been completed on the USB connection between the board and the PC. Bugs have been fixed that were caused by a clash between our encoding for transmitting switch status and communication control bytes used by the underlying layer.

We are going to use JUnit for the modular testing, and EasyMock for testing the GUI and some of the other classes that are used directly. Systems testing will be mainly via user interaction and systematic exploration of all features provided by the interface.

It is planned to test the Simulator extensively to ensure that it provides consistent results and correctly simulates sizeable circuits. We will verify the correct simulation using JUnit to make comparisons to the expected behaviour calculated by hand.

Correct Verilog output will be tested using JUnit for simple cases of single component Verilog output, but more complex situations will be analysed by hand and tested directly on the board to see that they correlate with the Simulated version.