Project Delta
An Interactive FPGA Circuit Simulator

Group 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 24, 2009

Contents

1 Overview **2**

2 Successes **2**

3 Failures **3**

4 Lessons learnt **3**

5 Summary of work undertaken **3**

5.1 Robert Duncan 3

5.2 Justus Matthiesen 4

5.3 David Weston 4

5.4 Christopher Wilson 5

5.5 Rubin Xu 5

1 Overview

The project has been a challenging experience for most of the members of the group, with pressing deadlines and all of the difficulties of working within a group.

Our system evolved from our original design due to technical issues with the implementation of Java using *JOP* on the *Altera DE2* board. The team worked well to work around the problem and create a new design that still involved communicating with the board.

This change meant that communication with the board was going to be continuous; sending packets back and forth between the board and the PC instead of a single bulk send of the circuit at the beginning of the simulation.

To compensate for the lack of ability to detach the board from the PC when simulating, Verilog exporting promoted from being an optional feature into a mainstay feature of the application.

After our first client meeting, we had new features to implement such as multiple languages, and a renewed focus on producing a polished UI due to the more run-of-the-mill nature of the bulk of the application. Our application now has a professional-looking GUI with features such as infinite zooming (due to our use of vector graphics for the components), undo and redo of changes, drag and drop of components, loading and saving of circuits, etc....

We found that some of our planned features were going to take too long to implement resulting in other features suffering or a severe overuse of our extremely limiting 60 hours per member time limit. We decided not to implement component grouping, though with the data structure we used it would have been relatively straight forward to implement should anybody want to use our software after the end of the project.

2 Successes

Our key success is a working, good looking multi-lingual application that simulates moderately complicated circuits.

On the whole the whole project was successful in terms of achieving the vast majority of our aims, as outlined in the specification, within the required timeframe.

Most of the team put in a lot of effort into produced code that was readable to other members of the team, and was well-designed. Extensive use was made of modular patterns, allowing other members of the team to use a predetermined interface. There was also a good deal of usage of common Java idioms such as subsumption, abstract classes, singletons, and factory patterns.

The use of Java has meant that producing a recognisable, easy to use, yet relatively straight forward GUI was easy using both standard java librarys in the `javax.swing` package and other open source libraries such as `JGraph`, `Swingx`, and `svgSalamander`.

Rubin's working in adapting the *JOP* software to our needs was excellent in quality and in timeliness, provided us with an abstracted interface to the hardware that enabled the rest of the team to get on with their own assigned areas within the project.

3 Failures

Communication was sometimes difficult with some members of the team due to delays answering emails, and late attendance to meetings. However, this was not a big issue as on the whole most of the team were quite interactive.

The application coding phase overran the initial time we had assumed it would take and of course it has made the last few weeks of the project much more intensive than previous weeks.

There was a failure to distribute the project evenly in terms of both members and the phase of the project. Some members of the team had large workloads towards the beginning of the project whilst others had large workloads at the very end of the project as we rushed to produce a finished project.

With *ad hoc* meetings, we failed to introduce a more formal sense to the project in its early stages. This meant that deadlines inevitably slipped and meetings juggled around.

We failed to include component grouping in the final product due to delays in other areas of the project. This was not a big loss though, as we had never put much emphasis on this feature and it was an optional feature if we found ourselves with time left over towards the end of the project.

4 Lessons learnt

Communication is extremely important in a group project. Poor communication between team members leads to confusion, missed deadlines and work being unevenly assigned.

Meeting targets is extremely important within a project on such a short time scale. One person's missed deadline can delay another person's work and cause a knock-on effect.

It is important to hold regular formal meetings with deadlines attached. Without these the project lacks structure and can be detrimental in motivating the team to succeed.

5 Summary of work undertaken

5.1 Robert Duncan

(a) Elected project manager, and performed most administrative functions.

(b) Set up SVN repository at *Google Code*.

(c) Produced Verilog translator that converts an existing circuit in our custom data structure into Verilog.

(d) Created custom translation package, along with French translation.

(e) High level board interface on the PC side.

(*f*) Added settings file for setting the language to be something other than default.

(*g*) Various GUI enhancements including: mnemonics, menu actions, SVG icons, ...

5.2 Justus Matthiesen

Justus' responsibility was to implement a suitable algorithm for digital circuit simulation, to provide the logical data structures (e.g. they do not store any visual information) necessary to represent the various gates and components, and circuits consisting of such components connected by wires. Furthermore, it was his task to implement (the logic side of) all gates and components mentioned in the technical specification. Summary of contributions:

(*a*) Looked into a number of algorithms for logical simulation and then implemented a discrete-time simulator that operates on the gate level and uses three-state logic. It is event-driven and assumes unit-time gate delays.

(*b*) For simulation purposes, he designed an abstract representation of gates and a data structure for circuits that only consist of gates and wire.

(*c*) Since the simulation data structure does not correspond to how the user views a circuit, he also implemented a higher-level data structure (based on components) which mediates between the users' view and the low-level simulation data structure. (It should be mentioned that Robert and Christopher helped to get rid of a number of issues this data structure initially had.)

(*d*) Together with Christopher, he ensured that the GUI circuit data structure is correctly translated into this higher-level data structure.

(*e*) He implemented the logical representation of all available gates and components.

(*f*) Wrote unit tests for most of the code he was written (still in the process).

(*g*) Various other contributions: Looked into libraries relevant to the project (e.g. graph libraries, SVG drawing libraries), designed a logo for the project, wrote a build script and has set up a build server which will hopefully prove helpful for regression testing in the last stage of the project.

5.3 David Weston

(*a*) Created the basic window layout with the necessary toolbars, menus and panels which integrated all the GUI classes.

(*b*) Created the component panel, with an abstract representation of categories and components, and a method of displaying them automatically with collapsible menus for each category, and mouseover effects.

(*c*) Created the clock panel, animation with adjustable speed and set this to run with the simulation.

(*d*) Implemented the interface for saving, loading and creating new circuits.

(*e*) Implemented printing.

(*f*) Tested the GUI.

5.4 Christopher Wilson

(*a*) Implemented classes to represent all the basic gates on the diagram

(*b*) Linked these with the underlying circuit diagram used for simulation

(*c*) Added ability to manually move wires around by adding and removing control points

(*d*) Imposed constraints on the wiring - no input-input or output-output connections, and only one wire per input

(*e*) Implemented classes for LEDs, 7-segment displays and switches

(*f*) Made it possible to choose which board component these represent by using a dialog box

(*g*) Imposed constraint that no LED or 7-segment display can be used twice in the same circuit

(*h*) Used port rendering to show component input/outputs as coloured blobs that are highlighted on mouseover

(*i*) Helped set up loading and saving of circuits using serialization of the diagram (and underlying circuit graph)

5.5 Rubin Xu

(*a*) SDRAM controller for JoP.

(*b*) Memory mapped I/O of LEDs, 7-Segment 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 in C.

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

(*f*) Unit testing of the communication channel.

(*g*) Chinese and Japanese UI translations.

(*h*) SVG Icon rendering from the open source project SVG Salamander.

(*i*) Structure of the help files.