Project Delta
An Interactive FPGA Circuit Simulator

Individual Report – David Weston

Contribution to Project

At the planning stage of the project I was assigned, along with Chris, to work on the graphical user interface. This initially involved deciding how we wanted the interface to work and how the main window would be laid out, as well as considerable research into the `JGraph` library which we decided to use for the circuit representation (at Justus's suggestion). I also wrote the description of the GUI included in our project specification.

After the first meeting with our client, it was decided that a good division of work would be for Chris to concentrate on the circuit representation with `JGraph`, dealing with manipulating components and wires in the main part of the window, while I should focus on everything else.

The first programming task I had to complete was setting up the window with a menu bar, toolbar, main panel for the circuit, and a side panel spit into the list of components and a little area to display the clock. This, and much of my later work, involved me reading a great deal of documentation about the Swing API; as a consequence of this project I now understand how Swing works much better than I did before.

Once I had got Eclipse working with Subversion and could access everyone else's work, I was able to integrate Chris's `CircuitPanel` class into the main panel of the window. I created buttons on the toolbar which performed some of the actions Chris had already implemented (such as zooming, copy and paste), as well as adding the same functionality to the menus. I should mention that at one point Rob stepped in to help clean up some of this code, and made it more consistent with Chris's.

For the `ComponentPanel`, the client suggested at our first meeting that it should be in a “concertina” style, with separate categories of components which could be expanded or contracted, rather than just having a flat list of everything in, requiring the user to scroll down an extensive list. I created some inner classes to represent this, with the `ComponentPanel` containing a list of categories and each category containing a list of components. I then created a simple interface to add categories to the panel, and to add components to each category, which abstracted away from the graphical representation (which I left till later to implement). So calling the constructor for `ComponentPanel.Category` automatically added the category to the list of categories (the `private ArrayList <Category> cats` field in `ComponentPanel`), and calling the method `void add (ImageIcon, String, int)` added a component to the category. This made changing the categories and components in the panel a whole lot easier, and people other than myself were able to do it later once they had implemented new components.

To draw the panel, I initially overrid the `paintComponent` method and used the `Graphics` class to draw images. After talking with other members of the group, I realised this wouldn't work with the drag-and-drop interface which Chris was implementing, so I switched to using icons arranged by a layout manager, which required me reading a lot

more documentation, but I think it produced a much better result in the end.

In order to create collapsible menus, I experimented with the `JXCollapsiblePane` class which Justus had found in an extension to Swing (the Swingx library). I had many problems getting this to work – the layout manager seemed to be ruining the effect. I read through some of the code in the class and realised that one way of showing and hiding part of the menu was simply to call the `setVisible` method, something which I could do myself without the help of an extra complex class. I eventually solved the problems I had been having with the layout manager (where categories kept expanding to fill the panel, though they were meant to be restricted in size) by adding a `Box.Filler` which I set to automatically change size depending on the size of everything else in the panel, to fill up any empty space. At this stage I spent a considerable amount of time debugging, playing around with different sized categories and resizing the panel and the window to try and find any problems – it took a while, but I seem to have fixed all (or at least the vast majority of) the bugs with it.

The panel looked rather dull, so I added some nice effects for when the mouse hovers over the category names, or the individual components.

The animated clock was another contribution of mine. I implemented it by subclassing `JLabel` so that hands would be drawn on the clock face, at angles determined by fields in the class. I then created the `ClockUpdater` class, which ran in a separate thread and decreased the angle of the hands (the hour hand being naturally 12 times as slow as the minute hand) by an amount relative to how long it had been since the last update (to provide a smooth animation no matter how the threads were scheduled). The speed of the hands was also proportional to the value in the `JSpinner` which I had put in the clock panel. The idea was for this to adjust the clock speed in the simulation.

Finally, once that appeared to be working I created methods for starting and stopping the clock animation, which I called in the `RunAction` and `StopAction` classes, so that the animation would only run while the circuit was being simulated. Connecting the `JSpinner` value to the simulation clock speed I left for others to implement.

One of my final major contributions was the interface for saving, loading, and creating new circuits. We had a lot of problems trying to serialise our circuit representation, but once this was fixed I was able to perfect the loading and saving mechanism, which only permits users to save circuits as “.cir” files, and prevents the user from loading anything other than a “.cir” file. Before loading a circuit or creating a new circuit, I decided it would be best to prompt the user to save their current circuit, in case they lost a lot of work by mistake.

At a suggestion from Rob, I also added a feature which allowed the user to print their circuit, which proved to be relatively straightforward to implement.

Throughout the project I did a fair amount of testing of the user interface, both my contributions and Chris's.

Evidence of contribution

The following classes are mainly or exclusively my own work:

`org.delta.gui.CircuitFileFiler`
`org.delta.gui.ClockLabel`
`org.delta.gui.ClockUpdater`
`org.delta.gui.ComponentPanel`

`org.delta.gui.NewAction`
`org.delta.gui.OpenAction`
`org.delta.gui.PrintAction`
`org.delta.gui.SaveAction`

The following class is a lot of my own work, but also has significant contributions by others:

`org.delta.gui.MainWindow`

I also made some contributions to the following classes:

`org.delta.gui.ComponentPanelLabel`
`org.delta.gui.RunAction`
`org.delta.gui.StopAction`

Contribution of other group members

Robert Duncan

After taking the lead in our first meeting, Rob was unanimously elected as project manager, and has done an excellent job. While we were planning he was diligent in working out everything that needed to be achieved, and throughout the project he has been very effective in assigning roles to people, and gently pushing us to complete our delegated tasks without ever showing anger or annoyance (even when we were ineffective).

As well as managing the rest of us and producing some excellent documents, he has managed to implement the entire Verilog module as well as many other features, such as the translation into different languages, some of the low-level interface between the computer and the DE2 board and linking this with the simulation. He has also been very good at tidying up all sorts of miscellaneous things, particularly in helping me and Chris with the GUI.

Justus Matthiesen

Justus was very good at doing his research during the planning stage into various algorithms related to his side of the project (the circuit simulation), as well as libraries for use with the GUI – without his idea of using JGraph, the whole project would have been a lot more difficult as we would have to implement a lot of functionality from scratch which JGraph had built-in. He has shown that he clearly knows what he is doing with the circuit simulation and has diligently produced a great deal of excellent code – admittedly little of which I have had any interaction with whatsoever, so cannot say a great deal about. He has been good at voicing his concerns to the group.

Christopher Wilson

Chris bravely decided to take on the circuit representation part of the GUI, which I am very thankful for as the JGraph documentation which I had been reading up to that point was rather overwhelming and I was not looking forward to putting it into practice. He managed to work out how to adapt the library to our own purposes, and has worked hard in getting the system to look good and work smoothly. He has also been very helpful when I have encountered problems.

Rubin Xu

Rubin worked extremely hard at the beginning of the project in researching options for use with the DE2 board, and then creating the interface with JOP. During the first two weeks he spent by far the most time on the project than any other group member, and I have been very impressed with what he has achieved, most of which is completely beyond my knowledge and capability. During the later stages of the project when he had less to do he willingly contributed to some areas which were clearly not his first choice (namely the documentation).