BELLATRIX

DEVELOPERS' MANUAL

by

ORION

Emin ÖZCAN

Mehmet Ergin SEYFE

Mehtap Ayfer PARLAK

Ilgın YARIMAĞAN

Eren YILMAZ

Table of Contents

1. Introduction

1.1 Project Definition

Bellatrix is a Digital Circuit Simulator tool which is capable of performing the simulation of digital circuits consisting of various wires and components such as multiplexers, flip flop, and gates etc... Bellatrix enables extended functionalities such as an advanced GUI, on-the-fly simulation, scripting support, option to export the circuit schema as a PDF file which are basically aimed to ease the jobs of students that are taking logic design laboratories. Users will be able to save a Bellatrix file in several formats such as ".jpg", ".pdf" and ".bx".

Bellatrix is mainly designed for educational purposes to introduce both the students and assistants of logic design courses with a program that provides not only the basic capabilities of a circuit simulator but also extended features like printing and scripting support etc. Besides, Bellatrix does all of these in a user friendly graphical environment. Using Bellatrix students will be able to generate their circuits and work on them easily and instructors can make evaluations more efficiently. Bellatrix is also useful for anyone studying digital circuits.

1.2 Document Scope

This document aims to give an idea of the whole system structure for future developers of this project.

After briefly describing some facilities of the Bellatrix, now we will present the system

architecture and other software related issues to help the software developers who want to develop our product. Then we give the architecture of our components one by one.

Finally we will give some development suggestions on some components.

We will now give an overview about the main modules the submodules of Bellatrix which will be explained in more detail in the following sections:

- GUI Management
  - Menu Management
  - Component Pane Management
  - Drawing Panel Management

- Script Console Management

- Draw &Design Circuit
  - Line & Component Design
  - Editing Drawing Panel

- Simulation Management
  - On-the-Fly Simulation
  - Manual Simulation

- Script Management
  - Load Circuit
  - Set Input/Output Values

- File Operations
  - Save Management
  - Load Management
  - Print Management(Export as Pdf)
  - Export as Jpg

2. GUI Management

2.1 Menu Management

Menu Management is waiting for incoming events from the user continuously all the time with MenuItem actionListeners.When an event is triggered by the user by some action, it is handled with the specified method associated with that event.

2.2. Component Pane Management

Component Pane Management also listens for upcoming actions of the user such as changing the tab between “Basic” and “Extended” or clicking on a specific gate button.

When the Component Pane Tab is changed from “Basic” to “Extended” for example the buttons in the Component Pane are changed from “Basic” buttons to “Extended” buttons.

When the user clicks the And Gate button for example, the and gate will be added to the current Drawing Panel with the next.

2.3 Drawing Panel Management

Bellatrix is a Multiple Document Interface. The user is able to add any number of sheets and draw different circuits at each sheet. The user can add or delete sheet at any time, also each sheet can be saved separately to a different “.bx”.

Each sheet actually generates a separate *DrawingPanel* object which will be added to the Drawing Panel vector *panels* of the GUI class and the user can work on these Drawing Panels concurrently simply by changing the tab number from “Sheet1” to “Sheet2” for example.

When Bellatrix is in “Add Component” or “Add Wire” mode (when a component button or the Wire button is clicked at the previous action) that object is added to the currently selected Drawing Panel at the next step.

2.4 Script Console Management

Script Console Management listens the script console to detect an enters action. When the user enters a proper command, the *JythonEngine* class performs the necessary operation.

3.Draw & Design Circuit

Draw Design Circuit performs the necessary operations initiated from the user by triggering events via the Menu Management.

3.1 Line &Component Design

There are two types of objects that can be added to a Drawing Panel.

The first one is a *ComponentGraphCell* object which has two types: Basic, and Extended. These are gates of the digital circuit. The second one is a *LineGraphCell* which represents the wires between these component Gates.

Basic Gates are as follows: And, Or, Not, Nand, Nor, Xor, Clock, InConnection, OutConnection, Ground, Vcc, Clock, Buffer

Extended Gates are as follows: Multiplexer 1*2, Multiplexer 2*4, Multiplexer 3*8, Decoder 1*2, Decoder 2*4, Decoder 3*8, J-K Flip Flop and T Flip Flop and D Flip Flop.

There are two hashtables storing the components and lines of each DrawingPanel object, named as *line_hashtable* and *comp_hashtable*. Each gate addition to a DrawingPanel constructs a ComponentGraphCell object and this object is added to the line_hashtable Hashtable. Similarly each line addition constructs a ComponentGraphCell object and this object is added to the comp_hashtable Hashtable.

3.2 Editing Drawing Panel

Editing Drawing Panel section consists of actions effecting the appearance of the whole DrawingPanel such as zoom in, zoom out, changing the glow mode etc.

Each Drawing Panel has its own options which can be changed separately.

For example when the ZoomIn or ZoomOut buttons are clicked or a Zoom option is choosen from the Menu, the *setScale* method of the currently selected DrawingPanel is called with the proper scale parameter.

4.Simulation Management

Bellatrix has two different options when performing a simulation.

The default mode is On-the-Fly simulation and the other option is Manual Simulation.

4.1.On-the-Fly Simulation

On-the-Fly simulation is handled by the Simulation Thread. When a change is detected from the *graphChanged* method of the DrawingPanel, *SimulationThread* object takes the control and resimulates the circuit by using *SimulationEngine*. On-the-fly simulation runs on background, it doesn't interfere with the Draw/DesignCircuit Management operations.

4.2. Manual Simulation

When the user chooses the manual simulation option from the GUI, the simulation options switches to manual simulation and therefore won't be performed at every change. Manual Simulation waits until the user clicks the "Run Simulation" button from the toolbar, when the button event is triggered Manual Simulation simulates the circuit using *SimulationEngine*.

5.Script Management

Script management supports basically two operations which are loading a circuit and setting values to the InConnection and OutConnection Gates.

5.1.Load Circuit

When the script console detects the command *open("filepath")* entered by the user, *JythonEngine* object performs the load operation of the previously saved file in the location stated by *filepath,* using the *exec* method call of its *PythonInterpreter* object.

5.2. Set Input/Output Values

Using Script command line user can also set values to the input and output connections if their labels are set before from the DrawingPanel.

When the script console detects the command *set(label, value)* entered by the user, (where *label* is the previously determined label name of the input or output connection and value is the value to be assigned) *JythonEngine* class sets the given value to this in/outconnection using the *exec()* method call of its *PythonInterpreter* object.

6.File Operations Management

Bellatrix provides four possible option as file operations

6.1.Save Management

Save operation is performed by serializing the *ComponentGraphCell* and *LineGraphCell* hashtables to a ".bx" file.

When the user chooses the save option from the menu or clicks the save button from the toolbar, a *FileChooser* appears that allows the user to select a location from his/her computer. The *line_hashtable* and *comp_hashtable* Hashtables are serialized to the location choosen by the user from.

6.2. Load Management

Load operation is performed by deserializing the hashtables from the ".bx" file to the current DrawingPanel.

When the user chooses the load option from the menu or clicks the load button from the toolbar, a *FileChooser*appears that allows the user to select a previously saved ".bx" file from his/her computer. After the file is choosen, the data in the file are deserialiazed to *line_hashtable* and *comp_hashtable* Hashtables of the current DrawingPanel objectin order to be used as the components and lines of that DrawingPanel.

6.3. Print Management (Export as Pdf)

Bellatrix enables the user to export the circuit drawing of each Drawing Panel to a printable pdf file individually. This way Bellatrix provides a print support for the circuit design.

When the user chooses the "Export as Pdf " option from the menu or clicks "Export as Pdf " button from the toolbar a *FileChooser* appears that allows the user to select a location to export the drawing.

After the location is specified, the *graph* object of the currently chosen DrawingPanel is saved to a *BufferedImage* object using the *getGraph* function call of the DrawingPanel object. Then this *BufferedImage* object is written to a temp file using a *FileOutputStream* and saved as a temp jpeg file using *createJPEGEncoder* function call of the *JPEGCodec* class.

Then the *PdfWriter* object (a class used for genareting PDf document in itext library) generates the resulting pdf document by using this temp jpeg file.

6.4. Export as JPG

Bellatrix also enables the user to export the circuit drawing of each DrawingPanel to a jpeg file individually.

When the user chooses the “Export as Jpg “ option from the menu a *FileChooser* appears that allows the user to select a location to export the drawing.

After the location is specified, the *graph* object of the currently chosen DrawingPanel is saved to a *BufferedImage* object using the *getGraph* function call of the DrawingPanel object. Then this *BufferedImage* object is written to a temp file using a *FileOutputStream* and saved as a the resulting jpeg file using *createJPEGEncoder* function call of the *JPEGCodec* class.

7 Extending Bellatrix

Bellatrix, the digital circuit simulator tool aims to provide an easy extensible, flexible architecture about which we will give some development suggestions on some modules and components at this section.

A developer can continue to work on Bellatrix and extend its functionalities by referring the documents of the used libraries(JHDL, Jgraph, jython, itext).

First of all, when a developer wants to add a component to Bellatrix, his/her main reference will be the JHDL library.

The JHDL library supports simulation of many gates. When a developer wants to add a new gate to Bellatrix this can be done by performing a reflection from the ***byucc.jhdl.Xilinx.Virtex.*** The developer can also develop these gates by using JHDL library class methods.

All the additional information about JHDL library, JDHL docs and information about how to use JDHL library can be founded at the official web site of JHDL at :

http://www.jhdl.org/

Jgraph library also gives flexibility to future developers of Bellatrix to extend the functionalities of circuit Draw/Design capabilities of the Bellatirx by simply developing new methods from the existing Jgraph functions.

All the additional information about Jgraph library, Jgraph docs and information about how to use Jgraph library can be founded at Jgraph home page:

http://www.jgraph.com/

A developer can extend the script functionalities that are supported by Bellatrix by examining *JytonEngine* class and the Jython language functionalities.

Also a developer can extend the functionality of exporting the circuit design as a different file format by examining the *exportAsPdf* and e*xportAsJpg* methods of the GUI class and develop other file format supports such as HTML or PS in a similar approach.