REQUIREMENT ANALYSIS REPORT

BELLATRIX

DIGITAL CIRCUIT SIMULATOR

Emin ÖZCAN - 1298090
Mehtap Ayfer PARLAK - 1347855
Mehmet Ergin SEYFE - 1298215
Ilgın YARIMAĞAN - 1409101
Eren YILMAZ - 1298470

Table of Contents

1 Introduction

1.1 Purpose

This document is the requirement analysis report of Bellatrix, the Digital Circuit Simulator project. In this document, the technological and environmental dependencies, functional requirements and structural models of Bellatrix are stated in detail.

1.2 Scope of the Project

Our new project, Bellatrix, is simply a Digital Circuit Simulator. 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 automated testing, 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. So if you are studying on logical circuits and need a virtual simulator Bellatrix is just what you need!

2 Target Audience Survey

2.1 Interview with İsmet Yalabık

We interviewed with İsmet Yalabık, who is the assistant of Ceng232 Logic Design course. We mainly asked him questions about Diglog which is currently being used in the course's laboratory assignments, and the testing tool to check and grade the submitted homework.

- What are the good sides of Diglog, and why it is being used in this course?
    - *In this course, we need a simple application that should do the basic digital design jobs. Diglog is simple. Moreover, this application supplies a great library of components, and this library includes the chips that we can supply in our laboratory courses. We can not force the students do their software simulations and hardware implementations in separated ways. Students know which components are available in the laboratory, and they*

can find those components in Diglog library.

- We know that Diglog is not liked so much by the students. What are the lacks of this application in your point of view?
  - *First, and most important, Diglog is not user friendly in any way. Users of Diglog easily go crazy. If we think more technically, the internal system is not so powerful I think. When I added for over 100 "from/to connections" into one of my projects, the application crashed. Thus, I had to draw all the lines as they are. More about the structure, Diglog does not fully support delays in components. Students see that there is delay in components when they are implementing their circuits in laboratory.*
- Well, now I switch to the testing tool. We know that you use a testing tool, or script, to check and grade the Ceng232 homework. How does this tool work?
  - *We supply the inputs, and set the expected outputs. The tool simulates the circuits in the submitted homework and prints out the results. It also supplies some useful statistical data to see whether the student cheated or not. This system worked fine this much, helped us reducing the time spent to evaluating homework and finding cheats.*
- Does this tool run Diglog to simulate the homework?
  - *No. It simulates the Diglog simulation in a way, but I don't know how it works. This tool was written by Erkin Bahçeci, and I only use it. I don't know its internal structure.*
- OK. So we will contact to Erkin Bahçeci for this tool's code, or working principle.
  - *Yes. He is in Texas now; you can contact him through e-mail. I think he will help.*
- Now, I will ask general questions about our project. In such an application, what must be included, in your opinion?
  - *If you intend this tool to be used in the Ceng232 course, or such similar courses, it does not need to be a super program. Do not try to implement things such as optimum placement of wires and gates. Having the components that are used in our laboratories is a must for such program. Additionally, it should give a very good printing support, preferably supporting color printing. Also, if you wish to support a testing tool, it is better to have a good statistical summary producer in that tool. This tool may create the statistics such as number of gates, number of connections, etc. Furthermore, you may supply page system to your application. Diglog enables us to use several pages to draw the circuit, so you better include this.*

2.2 User Survey

We organized an interview with three junior students from Middle East Technical University, Computer Engineering department who took Logic Design course last year. All of them used Diglog as a Digital Circuit Simulator in laboratory preliminaries and homework. As a potential user they helped us in determining our project features

Participants:

- Ela YILDIZER (E)
- Pınar SÖNMEZOCAK (P)
- Mechmet KEMİKLİ İSMAİL (M)

❖ **Q)** *Did you face with any difficulty when drawing components or wires?*

E: There is a place at the bottom of window. It is a place for putting most frequently used components. It does not work properly. For example, when you put output and input labels side by side they stick together. So you can't use them. Also drawing wires between gates is really time consuming.

P: I couldn't find the gates easily. They were categorized by numbers. Such as 7070, 7004... I couldn't find the true gate without looking at the lab manual. It was very difficult to draw wires. When you left click, it continues to draw until you right click.

M: It is easy to draw wires and components in Diglog.

❖ **Q)** *Did you face with any difficulty, a program bug or incapability of program when you simulate your design?*

E: When you draw a wire over a gate, program does not give any error message. However it does not work.

P: In sequential circuits, it is very difficult to use clock time generators. I always needed to cope with timing problems.

M: No.

❖ **Q)** *Did you need a hard copy of your design? How did you print-out your design?*

E: I had never printed my design.

P: I didn't need a hard copy.

M: I used hard copies of my designs in labs. I always used print-screen then print my design as a picture.

- ❖ **Q)** *Diglog has a CPU usage of almost 100%. Did you know it? What do you want to say about this unnecessary system usage?*

  E: Even I couldn't listen to music when Diglog is running.

  P: I only concentrated in Diglog when I used it. I didn't need to use any other application. So, it is not a big problem for me.

  M: I hate Diglog when I saw system usage.

- ❖ **Q)** *We will add a feature to our program, so user can also design using chips that is composed of gates same as the chips used in laboratories. What do you think about this feature? Is it useful?*

  E: It will be wonderful. Converting gate diagrams to chip diagrams by hand is very difficult. There are always mistakes because you can't control them efficiently.

  P: Unfortunately, it is a late feature for me. I have already passed the course. But it will be very useful for students who will take Logic Design course next semester.

  M: Perfect!

- ❖ **Q)** *Do you want to add anything positive or negative about Diglog?*

  E: The names of the gates are same as the numbers of chips used in laboratories. It was easy to find chips in labs.

  P: No.

  M: The states of wires are shown in Diglog Red, black and green lines which of them shows the states wires. It helped me very much in finding my design errors.

3 Literature Survey

3.1 Applications

The Literature Survey help us to specify the features of our project, prevents us from unnecessary repetition of previous mistakes and unwanted reinvention of available designs, algorithms, techniques, code libraries and applications. There are lots of applications in the market designed for digital circuit simulation. Therefore, we selected seven of these programs and examined them as a third user.

Selected Applications:

- Deeds
- DigitalWorks95
- Diglog
- KSimus
- MicroDev
- 5Spice
- Xilinx

3.1.1 Deeds (Digital Electronics Education and Design Suite)

Deeds is an application which offers a learning environment for digital electronics. It is conceived as a suite of simulators. The simulators contained in it cover combinational and sequential logic networks, finite state machine design, microcomputer interfacing and programming at assembly level. Deeds includes "Digital Circuit Simulator", "Finite State Machine Simulator" and "Micro Computer Emulator".

Finite State Machine Simulator" and "Micro Computer Emulator" are out of our project scope. Therefore only Digital Circuit Simulator can be an example for us, so we only evaluate "Digital Circuit Simulator" in this report.

Although its GUI is not very powerful, it is an easy to use program. Also it is designed for students like our project. So we consider it as an example application in our project.

Positive:

- Advantages of GUI:
  - *Good layout of drawing and tool fields*
  - *Copy , cut, paste, undo, move commands*
  - *Grouped components*
  - *Multiple selection of components*
  - *Easy to set properties of inputs, outputs and clock.*
  - *Zoom in and zoom out options*
  - *Single wire selection*
  - *Print and page setup support*
  - *Standard, component, and dockable toolbars*
  - *Distinguishable input and output toggles*
  - *Inputs and clock can be initialized*
  - *Clock frequency can be selected easily*
- Good range of components such as:
  - *Inputs and Outputs*
  - *Clocks*
  - *Boolean gates*
  - *Decoders, Encoders*
  - *Multiplexers, Demultiplexers*
  - *Flip-Flops*
  - *Registers*
  - *Counters*
- Good simulation performance
- Multi-level undo support
- Error Check List Window
- Simulation Modes
  - *Interactive Simulation Mode*

- *Timing Diagram Simulation Mode*

Negative:

- Disadvantages of GUI
  - *No properties option other than inputs, outputs and clock*
  - *Color of the drawing grid background is fixed*
  - *States of the inputs can not be distinguished easily*
  - *No smart line drawing*
  - *Can not remove wire joints*
  - *Components can not be rotated*
- Lacks of the simulation modes
  - *In the simulation mode, wire states are not shown in the drawing*
  - *No truth-table generation*
- No ability to add user defined components.
- No testing tool
- No print preview support
- No help support
- Length of the single wire can not be changed
- Runs only under Windows

3.1.2 Digital Works

Digital Works is a graphical design tool for constructing digital logic circuits and analyzing their behavior. It is free software that runs on Windows operating system. The easy-to-use graphical user interface, rich components, customizable macro options and powerful simulation tool of Digital Works can be a useful reference for us while designing and implementing our project.

Positive:

- Has a powerful GUI with the following characteristics:
  - *Toolbar supporting a user friendly graphical view*
  - *Toolbar of supporting both file operations and digital components*
  - *Smart wire drawing mechanism*
  - *Grid view option for easy circuit drawing*

- *Customizable wire color*
- *Single wire selection in deletion*
- *Ability to move the wired components easily with object selector tool*
- *Ability to rotate selected component*
- *Separate window for viewing the details of a macro*
- *Logic History window for recording the inputs or outputs of a seven segment display.*
- *Rich help menu*
- *Maximized and minimized view options for macros*
- *Error reporting when wiring components*

- Rich number of predefined components such as:
  - *Boolean gates*
  - *Tri-State device*
  - *RS,JK and D flip-flops*
  - *RAM and ROM*
  - *Switches*
  - *Sequence generators*
  - *Clock*
  - *Power Supply*
  - *Light Emitting Diode*
  - *Seven segment display*
- Interactive input for simulations
- Other advantages of Digital Works are such as:
  - *Customizable clock speed*
  - *Customizable input number for components (2,3,4)*
  - *Efficient use of annotations*
  - *Good simulation performance with pause option*
  - *File operations including a proper print mechanism*
  - *Ability to create macros*
  - *Ability to convert a circuit into a logic element by means of macros*

- It is a free software

Negative:

- Some disadvantages for the GUI of the program:
  - *No undo support*
  - *No multiple selection of components*
  - *No truth table view for any component*
  - *No customizable color option for components and drawing background*
  - *No single wire drawing*
  - *No single wire selection in coloring*
  - *No log window support*
  - *No log multiple sheets support (There are only reference pages for macros to examine in a detailed view.)*
  - *No zooming support for a specified part of the circuit*
- Some disadvantages for the simulation tool of the program:
  - *No truth-table view option which shows the values of the input and output*
  - *No time line view of selected components' status*
  - *When the simulation is run, wire states can not be examined only input and output values can be seen.*
- Other disadvantages for Digital Works are such as:
  - *Some components aren't added to the toolbar such as counters, they have to be added as a macro.*
  - *No test tool support*
  - *Shifting from wiring mode to adding a component mode is very time consuming compared to applications implemented in default wiring mode*
  - *Copy-Paste support is not implemented in free version*
- Runs only under the Microsoft Windows Operating system

3.1.3 Diglog

Diglog is a large system that designs and simulates circuits. It is an open source application which can run on both Linux and NT. It can be accepted as a good application by thinking of it's abound of components and supports. But, the lack of graphical interface support and user-friendliness make it a troublesome application. As a result, it can be a very good model for our project if we discover its good features and deficiencies.

Positive:

- It has a user interface with the following positive features:
  - *Easy accessing to the basic components, namely Boolean gates.*
  - *Editing support.*
  - *Zoom In-Zoom Out support.*
  - *Label support to input and output.*
  - *Multiple selections of components.*
  - *Free area for custom shortcut of new components.*
  - *Horizontal and vertical scroll support.*
  - *Harmony between the colors of background and foreground.*
  - *Showing the time.*
- It has a lot of components such as:
  - *Input Output Devices.*
  - *Control Devices.*
  - *Connection Gates.*
  - *Standard Digital Gates.*
  - *TTL Gates.*
  - *VLSI Transistors.*
  - *Multiplexers, Demultiplexers.*
  - *Counters.*
  - *Analog Components.*
- Good simulation performance with the glow peculiarity.
- Simulation can be controlled.
- Plotting support.
- Grid support.
- File saving and loading support.

- Help support with a help document file.
- Open-Source.

Negative:

- It has a user interface with the following negative features:
  - *Lack of graphical support.*
  - *Not user-friendly.*
  - *Incapable of wire drawing.*
  - *Shortcuts are specific. Incompatible with the generic shortcuts.*
  - *Difficulty of usage of mouse when accessing to the side menus.*
  - *Leaving the user ignorant about its most features. Lack of a quick help support.*
  - *Difficulty of accessing to new gates.*
  - *No replacing features of the gates.*
  - *Very simple moving. Lack of order after the moving.*
  - *Fixed colors of background and components.*
- Also, it has some negative features in the simulation mode:
  - *Wire states can not be obtained unless glow peculiarity is opened.*
  - *No truth-table generation*
- Lack of testing.
- Unnecessary system source consumption (100% of the CPU).

3.1.4 KSimus

KSimus is an application for simulating networks with Boolean and floating point data types. It is an open source project developed as a KDE application running on Linux machines. KSimus is a good application in the sense of user-friendliness and simulation capabilities. It will be a good example to our project.

Positive:

- Has a powerful GUI with the following characteristics:
  - *Good layout of drawing and tool fields*
  - *Grouped components*
  - *Log window*

- *Customizable drawing grid*
- *Copy-paste support*
- *Multiple selection of components*
- *Smart wire drawing and component reordering*
- *Distinguishable input and output toggles*
- *A separate user interface section that only shows the input and output, hindering wires and gates*

- Good range of components such as:
  - *Boolean gates*
  - *Pre-defined flip-flops*
  - *Counters*
  - *Multiplexers, demultiplexers*
  - *Delay elements*
  - *Tri-State gates*
  - *Floating point operators*
  - *Simulation control elements*
- Good simulation performance
- Simulation can be paused
- Multi-level undo support
- Open-Source

Negative:

- Although GUI is powerful, some of the following points are missing:
  - *No single wire selection*
  - *Usage of non-unique graphics for Boolean gates*
  - *"Delete" button does not work*
  - *No single wire drawing (may be considered as an error reduction feature)*
  - *Fixed color for drawing grid background*
  - *Replacing a gate with another one is really time consuming*
- In the simulation mode, there are some lacks:

    - *In the simulation mode, wire states are not shown in the drawing, only can be followed in the "watches" section*
    - *No time line view of selected components' status*
    - *No truth-table generation*
- No testing tool
- Runs only under Linux

3.1.5 MicroDev

MicroDev is an Integrated Development Environment for logic devices such as microcontrollers and microprocessors. MicroDev package is composed of two tools running together: an editor and a simulator.

Meditor is an IDE which is used to edit or build and debug the microcontroller program while the simulator is running. (Meditor will not be considered in this report since writing microcontroller programs is out of our project scope.)

MSim allows the user to draw a simplified schematic of the electronic board and to simulate it. MSim IDE is based on a Multiple Document Interface concept. Several sheets can be used to draw the schematics and simulate them. MicroDev has an applicable GUI and it is easy to learn and use. Although most of its advanced features like analog libraries are out of our project scope, it can help us to determine the main features of our project such as GUI properties and supported actions.

Positive:

- An applicable GUI with the following characteristics:
    - *Separated drawing and workspace fields*
    - *Grouped and alphabetically sorted libraries in a tree-view structure*
    - *An output window which is used to display information related to the Edit and Simulation modes*
    - *A menu bar allows to access all MSim features*
    - *A tool bar summarizes the most frequently used features*
    - *User can reach the Most Recent Files under Project menu*
    - *Component menu can display the properties of the selected component*
    - *Output, Workspace and Status Bar can be hidden by View menu*

- *User can add or remove several sheets*
- *Two different view style of sheets are available: Cascade and Tile*
- *Multiple selection of components*
- *Input and Output connection is available instead of line drawing between sheets or components*

- Several type of libraries such as:
  - *Pic Library*
  - *Analog Library*
  - *Display Library*
  - *Gate Library*
  - *Memory Library*
  - *Source Library*
  - *Switch Library*
- Simulation is started when the user wants by selecting simulation mode.
- User can run, suspend or reset the simulator
- The project is compiled and optimized when the user changes the mode to simulation
- Automatic insertion of the component names
- Highlighting the selected component name
- Undo and Redo actions

Negative:

- There are some missing properties about the GUI such as:
  - *Print, Print Preview and Print Setup are not yet implemented*
  - *Supported actions are very few which are only undo, redo and delete component*
  - *Drawing a gate and drawing a wire in order to link components can not be done in the same mode. Transition to one mode to another is really time consuming*
  - *Wire drawing is problematic*
  - *Deselection of the current component requires a right click on a free place otherwise, a left click will put another component of the selected type (this feature can be useful if it depends on the user's choice which means some user can like this feature on the other hand some of*

them does not.)

- In the simulation mode, there are some lacks:
  - *In the simulation mode, wire states are not shown in the drawing*
  - *No truth-table generation*
- No testing tool
- In spite of the variety of libraries, the numbers of logical components are still very insufficient in other words; there are many missing gates.

3.1.6 5Spice

5Spice is an application that provides user a graphical user interface that wraps around a traditional Spice simulation engine and presents a single application to the user. 5Spice has ability to design and simulate digital circuits but it is better in analog circuits. By examining this software we determined the library that we will use in our project. After examining 5Spice we abandon from SPICE language and decide to use HDL and select JHDL from HDL family.

Positive:

- Has a user friendly GUI with the following characteristics:
  - *Good layout of drawing and tool fields*
  - *Grouped components*
  - *Log window*
  - *Copy-Cut-Paste support*
  - *Multiple selection of components*
  - *Smart wire drawing and component reordering*
  - *Unique Icons for Gates*
  - *Distinguishable input and output toggles*
- Good simulation performance
- Multi-level undo support
- Generates of Spice Net List automatically
- Scientific details in analog circuit simulation

Negative:

- Although GUI is powerful, some of the following points are missing:
  - *Fixed color for drawing grid background*
  - *No Zoom In–Out Options*
  - *No customizable color option for components and drawing background*
  - *Incapable of giving labels to the inputs and outputs*
- Range of components for digital circuits is not enough:
  - *Boolean gates*
  - *Pre-defined flip-flops*
- In the simulation mode, there are some lacks:
  - *No time line view of selected components' status*
  - *No truth-table generation*
- No testing tool
- Runs only under Windows
- Not a freeware

3.1.7 Xilinx

Xilinx is the most popular digital circuit simulation tool in Electric and Electronics department. It is not only a very sophisticated application but also has many advanced features such as field programmable gate arrays. Also it supplies Verilog and VHDL language templates for writing HDL codes. Xilinx is a very good application both in terms of GUI properties and supported actions. It is a very sophisticated program and has a very good help support. Although Xilinx GUI and actions are very good examples for our project, most of the features of Xilinx are too advanced for us.

Positive:

- An advanced GUI with the following characteristics:
  - *Separated drawing and workspace fields*
  - *Console field is divided into four parts which are Console, Find in Files, Errors and Warnings*
  - *Component symbols are categorized and categories and symbols are shown in different*

panels

- *Snapshot, module and library views of a project can be selected*
- *A menu bar allows to access all Xilinx features*
- *A tool bar summarizes the most frequently used features*
- *User can reach the Most Recent Files under Project menu*
- *Symbol Info button can be used in order to display the properties of the selected component*
- *Symbols List can be filtered according to a symbol name*
- *Very advanced search tool*
- *User can create multiple windows*
- *Two different view style of windows are available: Cascade and Tile*
- *Multiple selection of components*
- *I/O markers, Bus Taps and Net names are available*
- *Different line styles (straight or bended) and some geometric shapes are included in the tool bar*
- *Print, Print Preview and Print Setup are supported*

- Wide range of components as categorized below:
  - *Arithmetic*
  - *Buffer*
  - *Clock Divider*
  - *Comparator*
  - *Counter*
  - *Decoder*
  - *Flip_Flop*
  - *General*
  - *IO*
  - *Latch*
  - *Logic*
  - *Mux*

    - *Shift_Register*
    - *Shifter*
- Simulation is started when the user wants by selecting simulation mode.
- User can restart, stop or run step by step the simulation
- The project is compiled when the user changes the mode to simulation and errors and warnings are displayed if there are any
- Automatic display of the component properties and all input/output information.
- Many actions are supported such as; Undo, Redo, Cut, Copy, Paste, Delete, Paste Special, Update Obsolete Symbol, Rename Selected, Mirror, Rotate, Align Instances, Align Text, Select All, Select and Clear, Select Objects, Find and Object Properties and so on
- User can add HDL codes by using both Verilog and VHDL language templates
- User can check schematics
- Help Support

Negative:

- Deselection of the current component requires a right click on a free place otherwise, a left click will put another component of the selected type (this feature can be useful if it depends on the user's choice which means some user can like this feature on the other hand some of them does not.)
- In the simulation mode, there are some lacks:
    - *In the simulation mode, wire states are not shown in the drawing*
    - *No truth-table generation*
- Not open source

3.2 Technology Research

In order to develop a good and powerful digital circuit simulator, we searched various technologies, ranging from general purpose programming languages to hardware description languages.

3.2.1 Base Programming Language & Development Environment

At the beginning of the project, we had several options for the language of our program. These languages would also determine the framework of our development environment. After some research, we found four possible languages and respective frameworks for these languages:

- C (GTK+)
- C++ (GTK+)
- C# (.NET)
- Java

Except C#, all these frameworks support cross-platform application development. Although there is a project, Mono, to support .NET development under Linux, it is not mature yet, therefore we eliminated C# and .NET first. C and C++ are powerful languages with GTK+ framework, but when we search a development environment for these languages, we could not find a good cross-platform development environment. Even if GTK+ is itself supporting cross-platform application development, without a good cross-platform development environment, testing in different platforms will be hard. Also, debugging and error handling in these languages are not easy. Thus we eliminated C and C++. We agreed on Java, which overcomes all these lacks with its own structure. Java is naturally cross-platform, even the default JDK tools have very good debugging and error handling support.

After selecting the language, we looked for a good development environment. There are some popular development environments for Java:

- NetBeans
- Eclipse
- JBuilder

We chose Eclipse with no doubt; it has all the functionality we will need in this application. The development environment we chose, Eclipse, can be run both under Windows and Linux, thus we will not spend time on porting our code from one platform to another.

3.2.2 Hardware Description Language

Representing hardware in software is not a simple task. We, as computer scientists, may think of the hardware as a collection of some logic structures, but this is not the case in real life. The circuit components are made of different types of semiconductors, and each of these behave different when timing is taken into consideration. The software should emulate the timing effects effectively. Some of these components may also run concurrently in some circuits, thus the software emulation should also supply this concurrency. This is done by defining these constraints separately in the language. These languages are called Hardware Description Languages. We researched some of the HDLs to ease our work in describing the hardware in software. These HDLs are:

- VHDL
- Verilog
- SPICE
- JHDL

VHDL

VHDL, which stands for VHSIC (Very High Speed Integrated Circuit) Hardware Description Language, is a powerful and standardized language. Its design purpose was to overcome the troubles such as describing circuits of enormous scale and managing such circuits. Its syntax is clear, but does not resemble to C in any way. We are all familiar with languages of C syntax, therefore, this is a minus for this language. When we searched VHDL parsers written in Java, we couldn't find any running application. Writing a parser to VHDL is not considerable here, so we dropped VHDL from our list.

Verilog

Verilog, Verifying Logic, is a rivaling language to VHDL. Verilog is as powerful as VHDL, and has a C-like syntax which is a plus for us. However, we came across the same problem that we did in VHDL, there are no Java parsers for Verilog. Again, we could not even intend to write a parser for Verilog in Java. Therefore, we also eliminated Verilog.

SPICE

SPICE is used commonly in Electrical and Electronics Engineering to simulate circuits. However, SPICE is a low level language that mostly suits to analog circuit simulation. Simulating a digital circuit in SPICE may give very realistic results, but requires very deep coding in implementation process. Moreover, we do not plan to support analog circuit simulation in our program, for these reasons, we eliminated SPICE too.

JHDL

JHDL is an experimental project which is being developed at Brigham Young University. JHDL is not a language, but just a method to describe circuit components and connections in Java language. JHDL meets all the constraints we look for; has a C-like syntax, high-level and compatible with Java. We consequently chose JHDL as our HDL in this project.

3.2.3 Scripting Language

Scripting will be one of the core components of our project. In order to implement scripting successfully, we chose Python, since it is the most powerful scripting language available. Python syntax is very similar to C syntax. Moreover, Python has been ported to Java platform under the name Jython. Jython has all the capabilities of Python, and runs under Java environment. For these reasons, we will use Jython as the scripting language in our project.

4 Project Requirements

4.1 External Interface Requirements

4.1.1 IDE Overview

1. Bellatrix IDE shall be based on a MDI concept (Multiple Document Interface). Several sheets shall be able to be used to draw the schematics and simulate them.
2. Project title header shall contain the name of the application and the current active sheet.
3. Menu bar shall allow accessing all system features of the application.
4. Tool bar shall contain the symbols of most frequently used features.
5. Workspace view shall display the components that are available.
6. Console view shall be used to display information related to the Edit and Simulation modes.
7. Status Bar shall display some interesting information like the current cursor position in file when the user edits a script file or the system clock.

4.1.2 Menu Bar

- Menu Bar shall contain the following menus:
    - Project
    - Edit
    - View
    - Component
    - Add
    - Simulation
    - Macro
    - Window
    - Help

4.1.2.1 Project Menu

- *New* option shall enable the user to create a new project.
- *Open* option shall enable the user to open an existing project.
- *Close* option shall enable the user to close the current project.
- *Save* option shall enable the user to save the current project.
- *Save as* option shall enable the user to save the current project with a different name.
- *Print* option shall enable the user to print the current project.
- *Print Preview* option shall enable the user to see the print format of the current project.
- *Print Setup* option shall enable the user to view and change the print settings
- *Most Recent Files* option shall enable the user to view and open the most recently used files
- *Set Drawing Area* option shall enable the user to view and change the settings of the Drawing Area.
- *Exit* option shall enable the user to exit from the program.

4.1.2.2 Edit Menu

- *Undo* action shall be supported until the last action before the user save the current project.
- *Redo* action shall be supported until the first undo action.
- *Cut* action shall be provided to the user to cut the selected item.
- *Copy* action shall be provided to the user to copy the selected item.
- *Paste* action shall be provided to the user to paste the last cut or copied item.
- *Select All* action shall be provided to the user to select all items in the current sheet.
- *Deselect* action shall be provided to the user to deselect a selected item.
- *Select Inverse* action shall be provided to the user to select all items except the selected item in the current sheet.
- *Group* action shall be provided to the user to group the selected items.
- *Ungroup* action shall be provided to the user to ungroup the selected group.
- *Select and Clear* action shall be provided to the user to clear the selected items.
- *Delete All* action shall be provided to the user to delete all items in the current sheet.

- *Preferences* option shall contain the background color and grid view options.

  - *Background color* shall provide a color palet to the user in order to set the background color of the drawing area. Default color shall be white.

  -*Show Grid* option shall provide a grid view.

4.1.2.3 View Menu

- *Add Sheet* option shall enable the user to insert a new sheet to the current project.
- *Remove Sheet* option shall enable the user to remove the current sheet from the project.
- *Console* option shall enable the user to show or hide the console view.
- *Workspace* option shall enable the user to show or hide the workspace view.
- *Status Bar* option shall enable the user to show or hide the status bar.
- Zoom options shall be listed such that *Fit to Window, Fit to Page, 50%, 75% …, Custom.*

4.1.2.4 Component Menu

- *Properties* option shall display the name, code and input/output information of the selected component.
- *Library* option shall display the library pages which contains all the components that are available as listed.
- *Find Component* option shall display a pop up window which contains a text box for the name of the searched component and enable the user to find it.

4.1.2.5 Add Menu

- Add Menu shall include the connection components in the first part:

  - Wire

  - Bus

  - In Connection

  - Out Connection
- *Wire* option shall enable the user to connect components with wires.
- *Bus* option shall enable the user to add bus connections.
- *In Connection* shall enable the user to add input connection instead of line drawing between sheets.

- *Out Connection* shall enable the user to add output connection instead of line drawing between sheets.
- Add Menu shall include Component and Instance Name in the second part.
- *Component* option shall display the basic components panel in the workspace.
- *Instance Name* option shall enable the user to insert the name of an instance when the user selects the instance.
- Add Menu shall include geometric shapes and text options in the third part:

  - Arc
  - Circle
  - Line
  - Rectangle
  - Text
- *Arc* option shall enable the user to draw bended lines.
- *Circle* option shall enable the user to draw circles.
- *Line* option shall enable the user to draw straight lines.
- *Rectangle* option shall enable the user to draw boxes.
- *Text* option shall enable the user to add text.

4.1.2.6 Simulation Menu

- *Run* command shall start the simulation of the current project. All the following commands in this menu shall be selectable after the simulation is started by run command.
- *Pause* command shall suspend the simulation of the current project.
- *Single Step* command shall perform the simulation step by step and shall show the internal steps if a synchronous input,like a counter or a flip-flop, is present.
- *Reset* command shall restart the simulator.
- *Stop* command shall exit from the simulation of the current project and return to the edit mode.
- The project shall be compiled when the user changes the mode to simulation.

4.1.2.7 Macro Menu

- *Record* command shall save the current project as a macro in order to support reusage of the drawing.
- *Load Macro* command shall load a previously recorded macro.
- *Script Editor* shall open a text file with the file extension .bx in order to enable the user to write or edit a script file. These scripts can be used for testing.
- *Execute* command shall execute the selected script file.

4.1.2.8 Window Menu

- Cascade option shall cascade the sheets of the current project.
- *Tile Horizontally* option shall tile the sheets of the current project horizontally.
- *Tile Vertically* option shall tile the sheets of the current project vertically.

4.1.2.9 Help Menu

- *About* option shall connect to our web site in order to give information about the Project Bellatrix.
- *Language Help* option shall display the tutorials about the scripting language.
- *Custom Help* option shall display the tutorials about the usage of the tool.

4.1.3 Tool Bar

- Tool bar shall contain symbols of the most frequently used actions as two lines namely; top tool bar and bottom tool bar.

4.1.3.1 Top Tool Bar

- Top tool bar shall contain the following symbols:

  - *New* icon shall be the short cut of the create new project operation.
  - *Open* icon shall be the short cut of the open an exiting project operation.
  - *Save* icon shall be the short cut of the save the current project operation.
  - *Undo* icon shall be the short cut of undo the last action.
  - *Redo* icon shall be the short cut of the redo the last undo action.
  - *Workspace* icon shall show/hide the workspace view.
  - *Console* icon shall show/hide the console view.

- *Print* icon shall be the short cut of the print the current project.
- *Print Preview* icon shall provide the preview of the current project.
- *Zoom* icon shall provide zoom in/out options.
- *About* icon shall be the short cut for connecting to our web page.

4.1.3.2 Bottom Tool Bar

- Bottom tool bar shall contain the following symbols:

  - *Run Simulation* icon shall be the short cut for starting the simulation of the current project.
  - *Link* box shall be a combo box which contains the options: Wire, Bus, In Connection, Out Connection in this order. Wire shall be the default value.
  - *Arc, Line, Circle, Rectangle* symbols shall be the short cuts for the geometric shapes in the add menu
  - *Hand* icon shall enter the hand mode in which editing is not allowed.

4.1.4 Workspace

- Workspace view shall enable the user to see all available components.
- Workspace shall be divided into three sub panels: Basic Components, Extended Components, Custom Panel. The default panel shall be Basic Components.
- Basic Components panel shall include the most frequently used components such as and, or, nand, nor, counter, multiplexer and so on.
- Extended Components panel shall include the extended components.
- Custom Panel shall enable the user to store and save the components that he/she wants.
- A component shall be added to the Custom Panel by dragging the selected component to the custom panel area and dropping it.
- A search engine shall be available in the workspace in order to enable the user to find a component by its name or code.
- An orientation combo box shall be available in the workspace in order to enable the user to rotate or mirror the selected component.
- Orientation combo box shall contain the options:

  - Rotate 0
  - Rotate 90

- Rotate 180
- Rotate 270
- Mirror

4.1.5 Drawing Area

- Drawing area shall display the current sheet of the current project.
- All sheet names shall be displayed as tabs under the drawing area and these tabs shall enable the user to traverse between sheets.
- Drawing area shall be displayed as grid if the show grid option is selected by the user.
- Drawing area settings shall be able to change by the user by using the project menu.

4.1.6 Console

- Console view shall be used to display the information contained in the "console", "warnings", "errors", and "find in files" sheets.
- Console view shall display the warnings (if there are any) after the simulation of a project when the *Warnings* tab is selected.
- Console view shall display the errors (if there are any) after the simulation of a project when the *Errors* tab is selected.
- Console view shall display a command prompt in order to enable the user to execute commands by using this terminal when the *Console* tab is selected.
- Console view shall enable the user to execute a script file from the terminal when the *Console* tab is selected.
- Console view shall display results after the execution of a script file when the *Console* tab is selected.
- Console view shall display *Find in Files* sheet in order to enable the user to search for a specific word in all BX files.

4.1.7 Status Bar

- Status Bar shall display the cursor information when the user edits a script file.
- Status Bar shall display the mode or status information such as edit, simulation, executing, ready, etc...
- Status Bar shall display the system clock.

4.2 Non-Functional Requirements

4.2.1 User-friendliness

Our primary goal in this project is to create the best user interface among the other digital circuit simulators. Our research in the current product range showed us that the user interface of these programs does not satisfy the users' needs. We will create a powerful interface that can be modified according to user preferences. This way, the user can create the optimum workspace for his/her project.

The interface items will be well-defined that the user will easily access to everything without searching it deeply. The menus and toolbars will be clear, and the user will be able to access almost every menu item with either hotkeys or script console. Using the script console, the user will be able to use the application without a need to a mouse.

The drawing part will be designed in a smart way. User will not spend his/her valuable time on rearranging the wires or components. In most of the digital circuit simulators, user spends more time on the circuit's appearance than designing it. This problem will be overcome in our project, by applying smart drawing algorithms on the circuit.

4.2.2 Stability

Circuit design is not an easy task. It requires careful attention in both designing it logically and drawing on computer. We are not interested in logical design of circuits, but drawing it on the computer. If an unstable application is used to draw the circuit, it has no advantage to draw on computer. Therefore, the interface should be reliable, and has to have recovery abilities in case of a crash.

Additionally, the simulator of the application should be stable, especially in precise calculations. Some circuits rely on the well known characteristics of digital chips. The simulator should support all of these characteristics and obey the industry conventions. The simulator should realize the circuit virtually in every aspect.

4.2.3 Performance

Diglog application, the most widely used digital simulator used in education, is infamous of its CPU usage. That is not because of the complexity of algorithms; the wrong programming methodologies used in the application cause this problem. Bellatrix shall not fall into this problem. The system resource usage will be minimized to increase performance on slower computers. The user should be able to run other applications properly while Bellatrix is running.

4.2.4 Portability

Digital Circuit Simulation is a somewhat general topic. It is generally related with the low level abstraction of computers. Therefore, it should be able done in almost every computer platform. Bellatrix will have high portability, since it will be developed in Java. The application shall run in almost all computers having a Java runtime environment.

4.3 Software Requirements

4.3.1 For the Development Phase

Project Bellatrix will be developed in Java. Thus, Java Runtime Environment and Java SDK are essential tools to be used in the project. However, we will not use only the Java SDK. As a development environment, we chose Eclipse, an open-source application development environment for Java.

We intend that our project will be a cross-platform application. To achieve this goal, we will require at least two different platforms. One of these platforms is the Windows platform, the other one is the Linux platform. To be specific, the Windows environment will be Windows XP, and the Linux environment will be Fedora Core 4 (Linux 2.6 kernel) in the development phase. Other platforms will not be supported officially; however, the application is intended to work on any platform having a compatible Java Runtime Environment.

4.3.2 for the end user

Since the application is to be developed in Java, the end user will only need a compatible operating system (Windows XP or Linux 2.6), and a Java Runtime Environment. We will officially support only Sun's Java Runtime Environment, but support to Microsoft Java Virtual Machine may be added later.

At the end of the project, Bellatrix will be packed and distributed as a single setup file. These setups will not require any pre-installed software or framework. The Java Runtime Environment will not be distributed with the application.

5 Data Models

5.1 Data Dictionary

Name	Input Data
From	User
To	Graphical Interface
Format	text/stream
Description	The user inputs some data in the user interface, using the keyboard.

Name	Interface Commands
From	User
To	Graphical Interface
Format	action/mouse, action/keyboard
Description	The user commands the Graphical Interface using the mouse or the keyboard using hotkeys.

Name	User Script Commands
From	User
To	Script Engine
Format	text/stream
Description	The user commands written to the script console.

Name	User Script Data
From	User
To	Script Engine
Format	text/stream
Description	The user data given through the script console.

Name	Input File
From	File Directory
To	Input File Manager
Format	file
Description	The input file to the system. This file can be a Bellatrix format, .bx file, Diglog format .lgf file, or script file. Script file is any extension ANSI text file.

Name	Input Script File
From	Input File Manager
To	Script Engine
Format	file
Description	The script file taken from file directory to be processed by the Script Engine.

Name	Circuit Data
From	Input File Manager
To	Circuit Design
Format	internal structure/JHDL
Description	The input file is processed and converted to internal representation of circuit, in the JHDL style

Name	Circuit Data
From	Circuit Design
To	Output File Manager
Format	internal structure/JHDL
Description	The newly created/modified circuit is passed to the Output File Manager to be processed and converted to Bellatrix .bx format or Diglog .lgf format.

Name	Design & Draw
From	Graphical Interface
To	Circuit Design
Format	internal structure/unclassified
Description	The user actions are passed to circuit design are grouped under this name. These actions include mouse actions and keyboard actions.

Name	Preview & Change Settings
From	Circuit Design
To	Print Manager
Format	metadata
Description	This is the meta data of the circuit (page layout, colors, size) passed to the print manager.

Name	Print Information
From	Print Manager
To	Printer
Format	PostScript
Description	This is the meta data of the circuit (page layout, colors, size) passed to the print manager. This is for direct printing support.

Name	Execute Macro
From	Script Engine
To	Circuit Design
Format	script
Description	Pre-defined macros are executed and the generated data flows to the circuit design process. All other control data from macros flow over the circuit design.

Name	Record Macro
From	Circuit Design
To	Script Manager
Format	script
Description	The actions of the user recorded in circuit design is passed to script manager to be recorded.

Name	Output Script File
From	Script Manager
To	Output File Manager
Format	script
Description	The script is written to the disk.

Name	Simulate
From	Circuit Design
To	Simulator
Format	internal structure/unclassified
Description	This data flow contains the circuit information and simulation controls. The only access to simulation is over this data flow.

Name	Simulation Results
From	Simulator
To	User
Format	internal structure/unclassified
Description	The simulation results are ready to be shown to the user. These results include the logical output and statistical output.

Name	Output File
From	Output File Manager
To	File Directory
Format	Bellatrix .bx format/Diglog .lgf format/Printable file format(PDF/PS)
Description	The circuit data is converted to one of the formats of User's choice and written to the disk.

Name	Sheet Data
From	Interface I/O Manager
To	Sheet Control
Format	internal structure/unclassified
Description	This is the sheet configuration data.

Name	View Data
From	Interface I/O Manager
To	View Manager
Format	internal structure/unclassified
Description	This is the view configuration data.

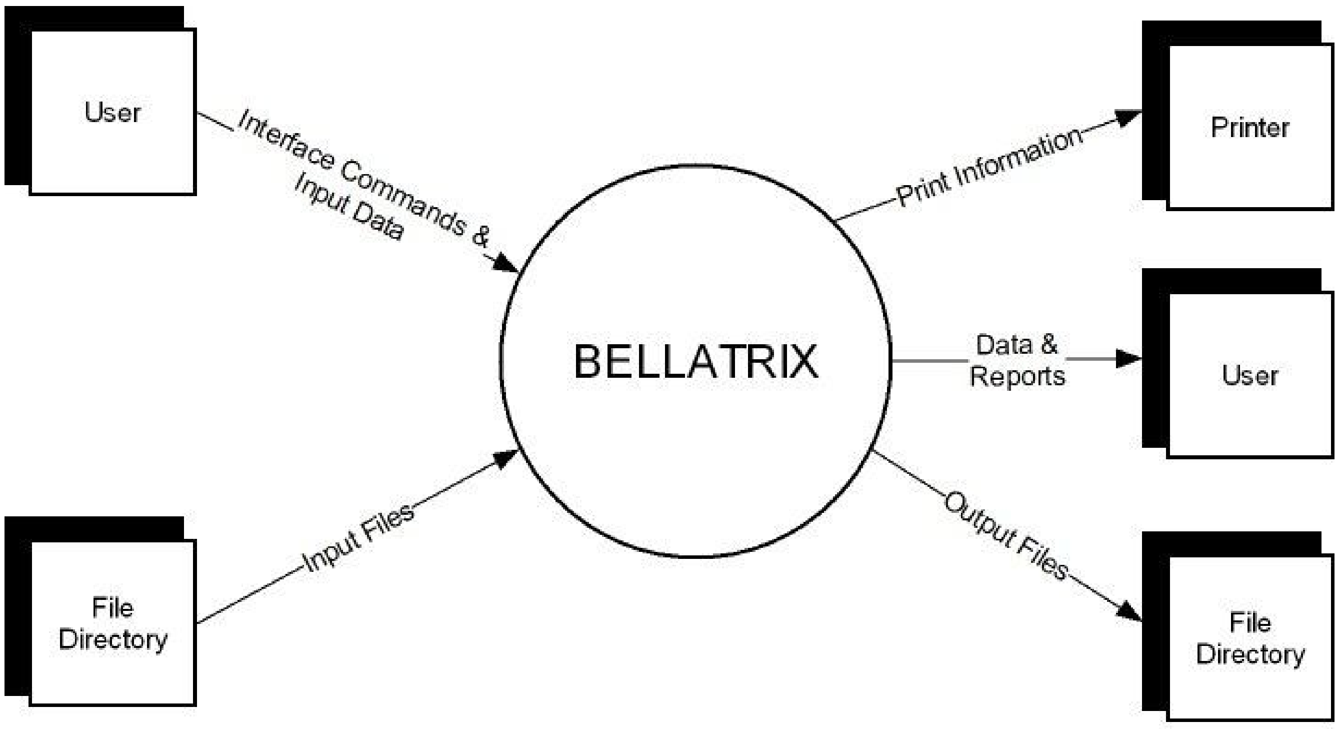
Name	Custom Panel Data
From	Interface I/O Manager
To	Custom Panel Manager
Format	internal structure/unclassified
Description	This is the custom panel configuration data.

Name	Search Data
From	Custom Panel Manager
To	Component Library
Format	text
Description	The searched component's data is transferred to the component library.

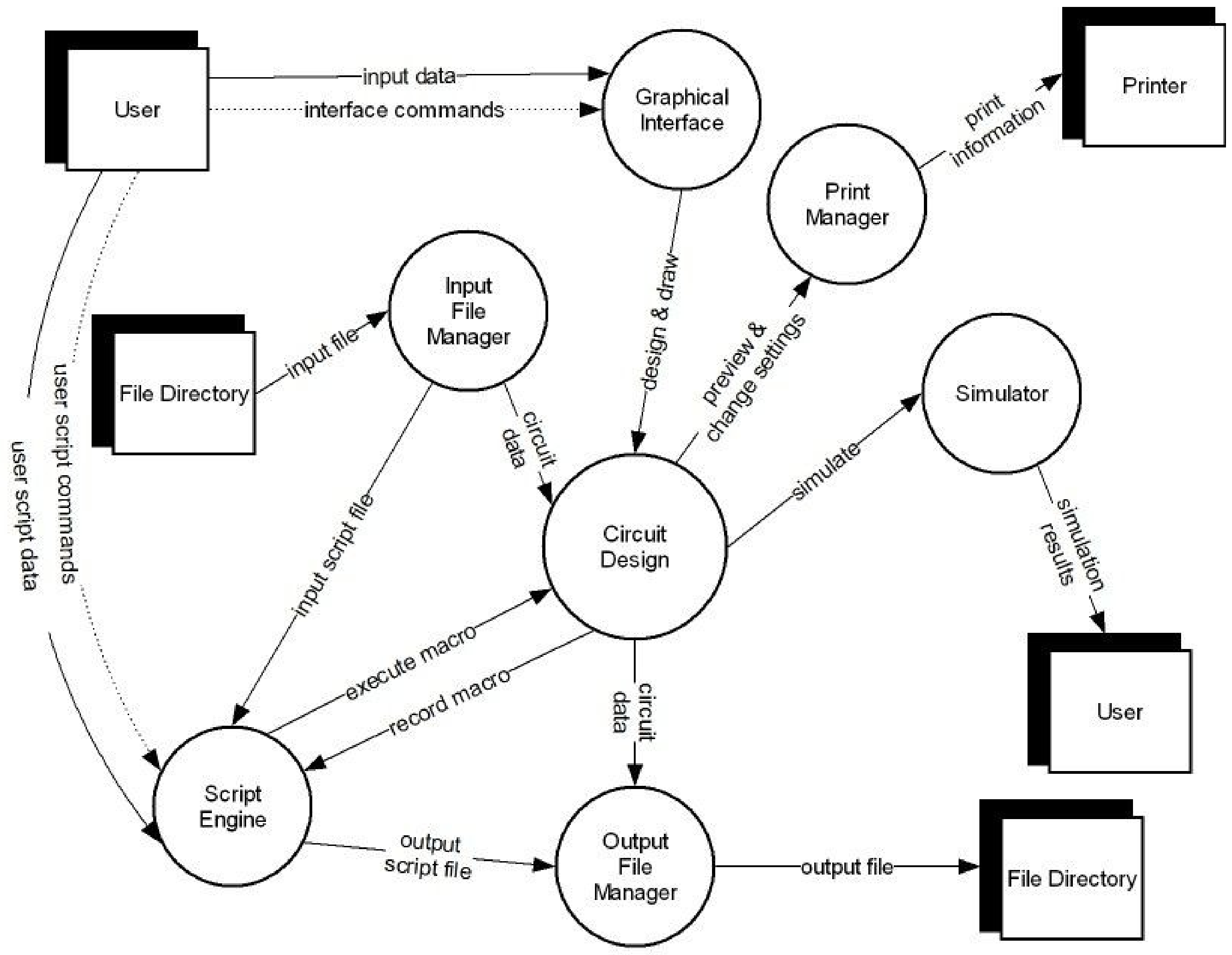
Name	Component Data
From	Component Library
To	Custom Panel Manager
Format	internal structure/unclassified
Description	The results of the component search is returned to the custom panel.

5.2 Data Flow Diagrams

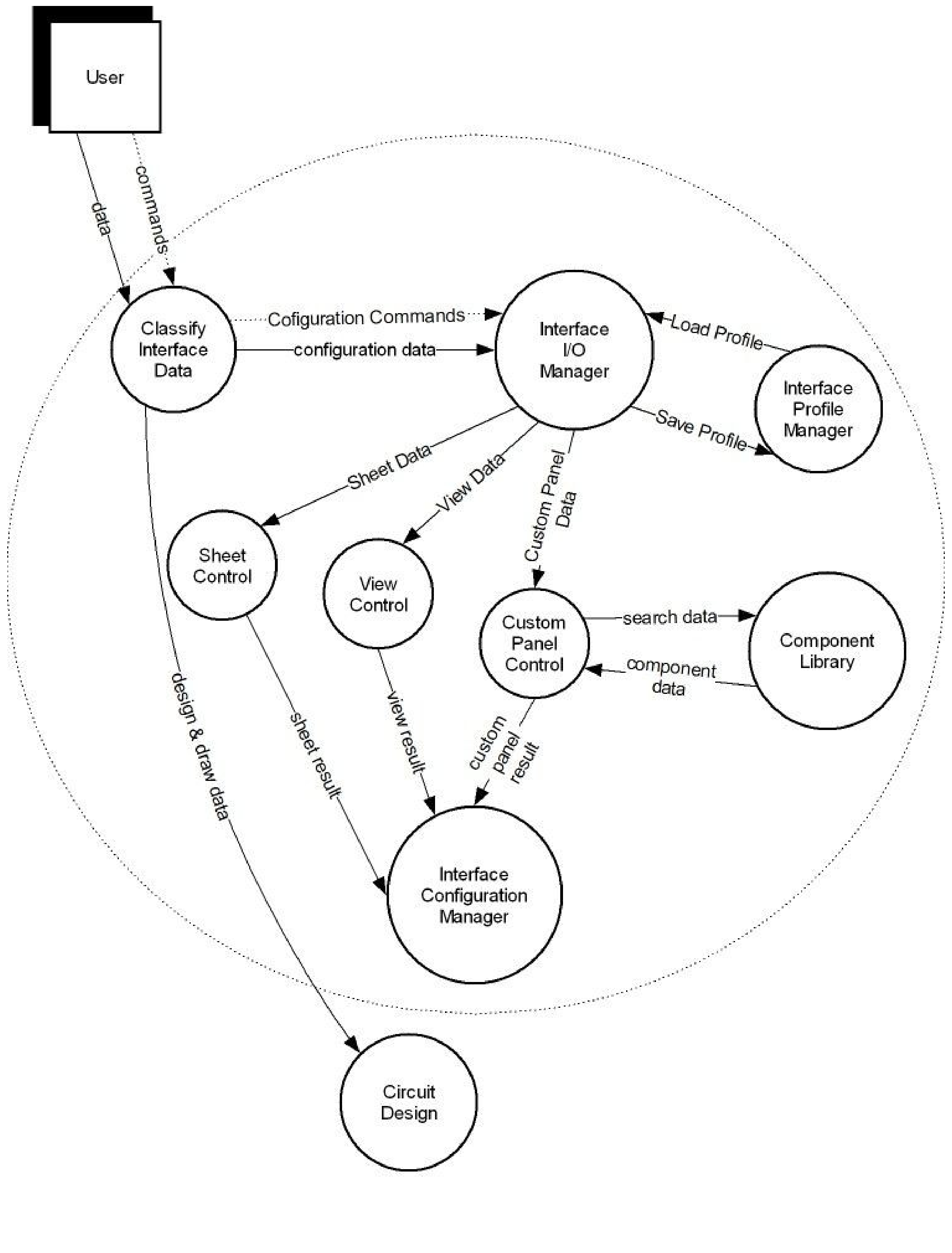
5.2.1 Level 0 DFD



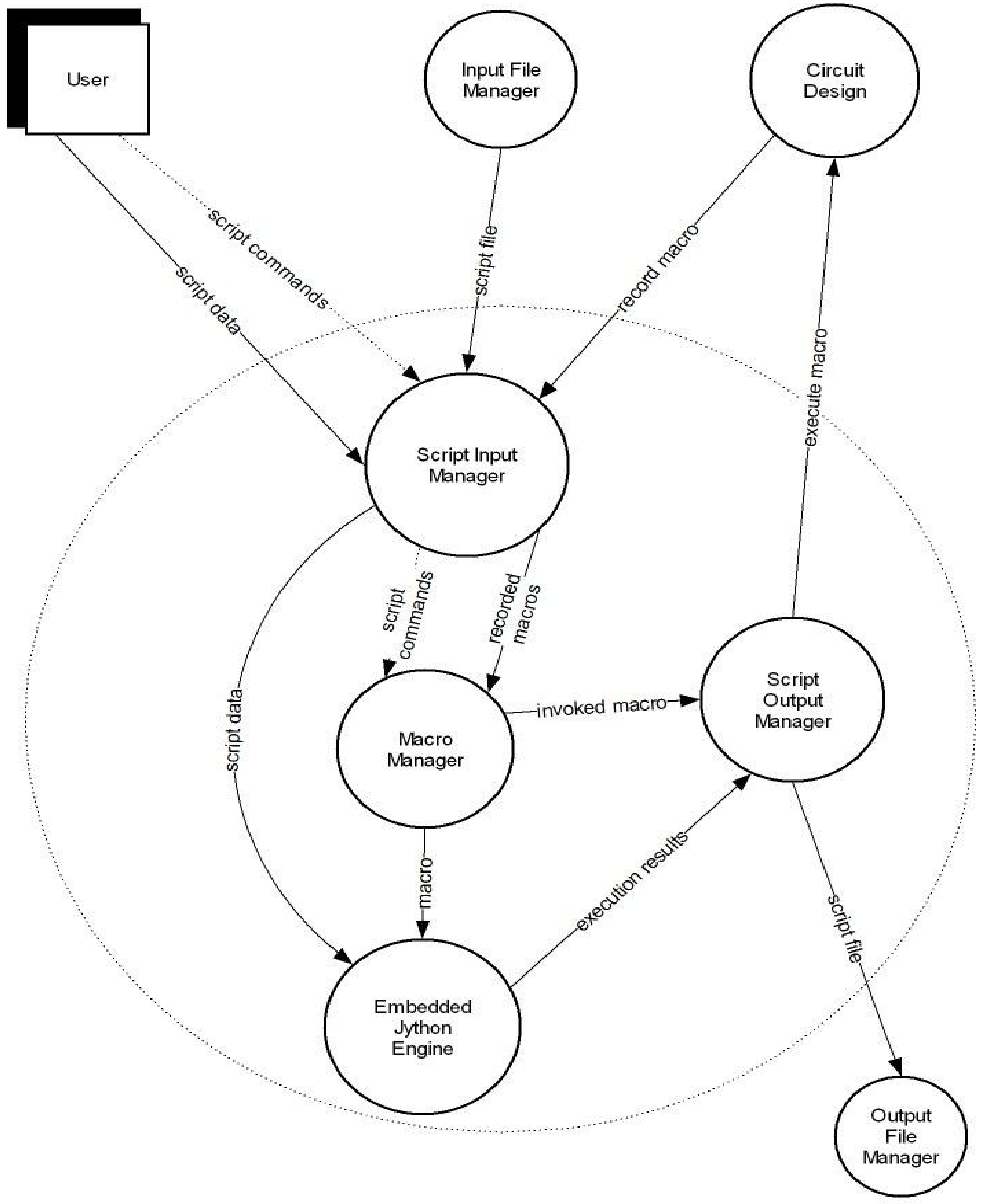
5.2.2 Level 1 DFD



5.2.3 Level 2 DFD – Graphical Interface



5.2.4 Level 2 DFD – Script Engine



6 Usage Modeling

6.1 Usage Scenario

In Project Bellatrix, user can design a circuit by opening a project. He can save the project at any time he wants, and later he can open the same project from the disk. If he wants to open another project, he has to close the recent one.

In a project, there are three spaces. First of them is drawing area. Drawing area is composed of sheets. User can open numbers of sheets and can remove some of them from the drawing area if he wants. Also he can operate these sheets view. He can choose tile or cascade view for seeing the sheets more easily. In project, there is also a workspace area that is used for showing and searching the components. Any user can reach any basic or extended components from there. Also he can create his custom panel by searching the required components from the component library. In addition to these spaces, there is also a console which is used for giving external controller code.

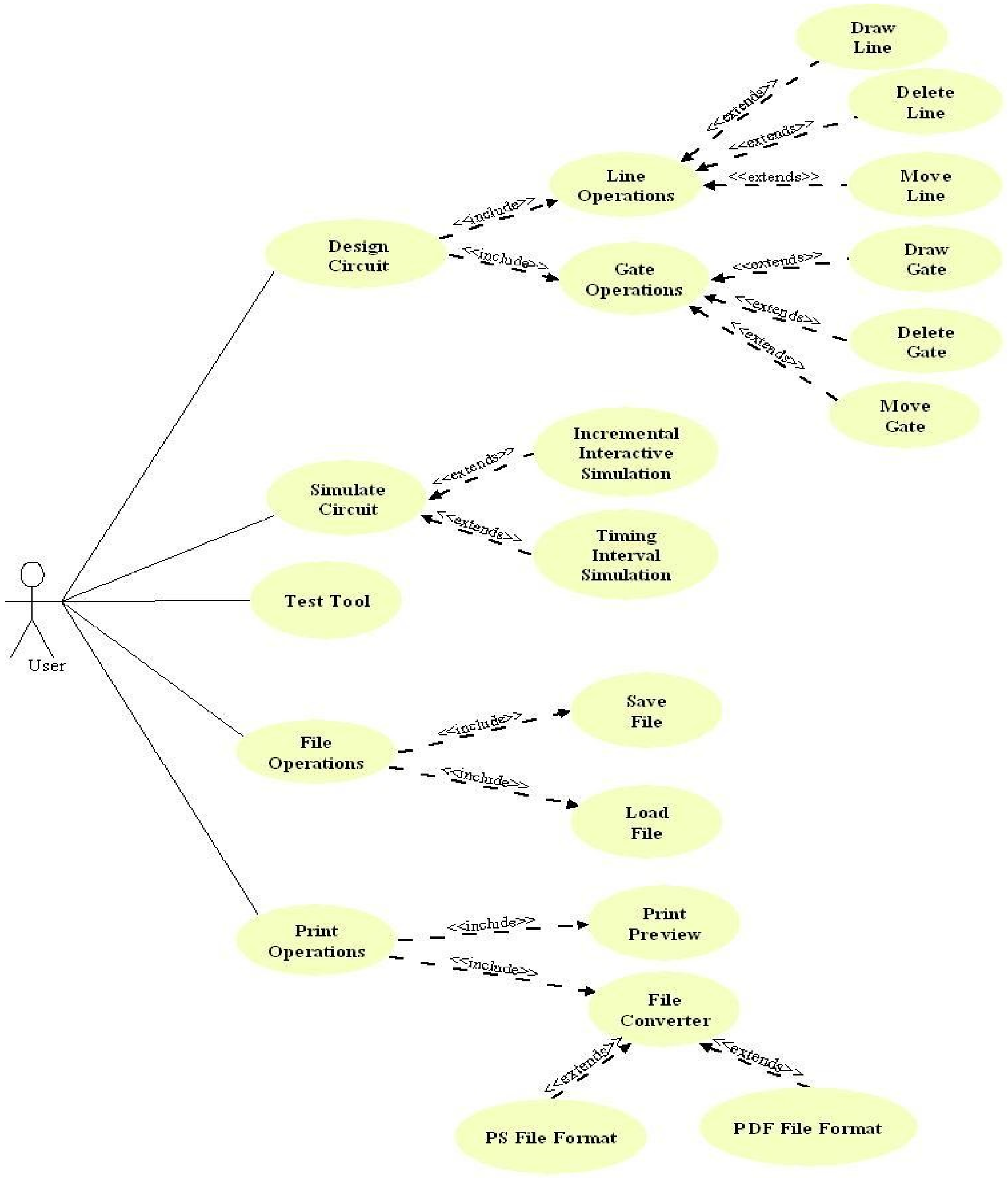
User can design a circuit by using drawing area. He can choose any component from the workspace menu and drag it to the drawing area. At any time,he can move the component,rotate the component by right angles, and copy, cut, paste and delete the component at any time. User can select only one or lots of components and he can group them. There is also a redo, undo and clear support. For each component, user can see its properties and he can change its color and name. In addition to component drawing, user can draw connection lines. Firstly, he has to choose the line mode from the tool bar for activating the line drawing. After the activation of line drawing mode, he can draw wires. This activation prevents the accidental line drawing. Drawing area has default view settings. However, any user could want to change these settings. For this reason, system provides some competence to the user, such as changing the background color and controlling the grid view. Also there is a zooming support. After doing some changes on the view of drawing area, user can save them and load them any time he wants. Finally, user may want to deactivate the drawing area. So, there is hand mode support. In hand mode,user can only see the circuit ,namely, he can do nothing to the circuit.

In our program, user can test the circuit by using the simulator. Simulation mode is inactive when the program is in edit mode. If a user wants to simulate the circuit, he has to switch on simulation mode by selecting Run, from the tool bar or from Simulation on the menu bar. Also he can start the simulation by giving a script code from the console. Then he can pause, reset or stop the simulation, again by using the tool bar or console. In simulation of a circuit, a user can give inputs once or step by step.

Project Bellatrix supports the user with a macro peculiarity. User can record the drawing process by using the menu bar, or by giving an external code from the console. For recording the process, our system creates a script file. The user can load this file and execute the macro for watching it. In addition the automatic script file creation, our system gives a chance of manual script file creation to the user. User can open a script editor from Macro on the menu bar and he can write his script code here. After loading and executing his own script file, he will watch what this script does.

Finally, our system has a print support. After drawing a circuit, user can print out this circuit on a paper. If he wants to see the view of the paper before getting the print-out, he has to select the print preview button from the tool bar. User could want to change default print settings. Therefore, after clicking the print button on the tool bar, a new pop-up menu will be opened and user can change the settings,such as paper size or color.

6.2 Use-Case Diagram



7 Behavioral Model

7.1 Events

Events provide stimuli to the state charts which produce the behavior. All events referenced by state charts are defined below.

- **Run_Simulation:** Press of the run simulation option from the simulation menu.
- **Stop_Simulation:** Press of the stop option from the simulation menu.
- **Library_Selection:** Press of the library option from the components menu.
- **Component_Selection:** Selection of a component from the library or from one of the component panels.
- **Link_Selection:** Press of a connection from the add menu or the tool bar.
- **Hand_Icon:** Press of the hand icon from the tool bar.
- **Open_Editor:** Press of the script editor option from the macro menu.
- **Close_Editor:** Closing the text editor.
- **Execute_Script:** Executing the script file from the simulation menu.

7.2 Modes

Bellatrix has three functional modes: An Edit mode with two sub modes, a Simulation mode and a Script Editor mode.

7.2.1 Edit

Edit mode is where the user may build or edit the schema by using the menus in the menu bar, adding components from workspace and linking these components together.

Entry:

- On entry to Edit mode the state shall transition to Edit::Design.

Behavior:

- In Edit mode Run_Simulation shall cause a transition to Simulation mode.
- In Edit mode Open_Editor shall cause a transition to Script Editor mode.

7.2.2 Edit::Design

Edit::Design mode is where the user is allowed to select, to place and move the components and connections.

Behavior:

- In Edit::Design mode Component_Selection shall cause a transition to Edit::Design::Component mode.
- In Edit::Design mode Link_Selection shall cause a transition to Edit::Design::Connection mode.
- In Edit::Design mode Library_Selection shall cause a transition to Edit::Design::Library mode.
- In Edit::Design mode Hand_Icon shall cause a transition to Edit::Hand mode.

7.2.3 Edit::Design::Component

Edit::Design::Component mode is where the user is allowed to select, to place and move the components.

Behavior:

- In Edit::Design::Component mode Link_Selection shall cause a transition to Edit::Design::Connection mode.
- In Edit::Design::Component mode Library_Selection shall cause a transition to Edit::Design::Library mode.

Selection

- In Edit::Design::Component mode selection of a component shall require the following steps:

1. Open one of the panels displayed in the workspace view or select the component option in the add menu.
2. Left click on the selected component
3. Choose the sheet to insert the component to (if not yet selected)
4. Left click to put the component. The component name will appear in red if selected.
5. Right click on a free place to deselect the current component

- A left click on the sheet on a free place will put an other component of the selected type.
- Several components can be selected by drawing a rectangle while the mouse left button is pushed.

Placement

- Left click on a component while the mouse is moved will move the component(s).

7.2.4 Edit::Design::Connection

Edit::Design::Connection mode is where the user is allowed to link the components together.

Behavior:

- In Edit::Design::Connection mode Component_Selection shall cause a transition to Edit::Design::Component mode.
- In Edit::Design::Connection mode Library_Selection shall cause a transition to Edit::Design::Library mode.

Wire Link

- In Edit::Design::Connection mode wires shall be used to link two components of the same sheet.
- In Edit::Design::Connection mode wire drawing procedure shall require the following steps:

1. Select the link icon in the tool bar or the add menu
2. Select Wire
3. Left click on the first component pin
4. Left click on the second component pin.

- Once the wire drawn, it shall be able to be moved or modified by selecting it.
- A wire shall be highlighted if the wire is connected to a pin.

In / Out Connection

- In Edit::Design::Connection mode instead of wires, In / Out connections shall be used to connect two components between sheets or within the same sheet without drawing a wire.
- In Edit::Design::Connection mode the In / Out connections are bidirectional. The type (In or Out) shall be used to recognize the signal direction.
- In Edit::Design::Connection mode wire drawing procedure shall require the following steps:

1. Select the link icon in the tool bar or the add menu
2. Select Out Connection
3. Left click on an empty place
4. Change the name of the connection (for example: clock).

5. Connect the Out Connection to a component pin with a wire.
6. Select In Connection
7. Left click on an empty place
8. Change the name of the connection (for example: clock).
9. Connect the In Connection to a component pin with a wire.
10. The two components are connected together.

7.2.5 Edit::Design::Library

Edit::Design::Library mode is where the user is able to see all available components from the library pages.

Behavior:

- In Edit::Design::Library mode Component_Selection shall cause a transition to Edit::Design::Component mode.
- In Edit::Design::Library mode Link_Selection shall cause a transition to Edit::Design::Connection mode.

7.2.6 Edit::Hand

Edit::Hand mode is where the user is only allowed to view the schema and is not allowed to edit by using the mouse clicks. This mode shall prevent the user to draw a wire or insert component accidentally.

Behavior:

- In Edit::Hand mode Hand_Icon twice shall cause a transition to Edit::Design mode.
- In Edit::Hand mode mouse clicks shall not do anything only the scrolling with mouse shall be enabled.

7.2.7 Simulation

Simulation mode is where the circuit simulation takes place.

Entry:

- On entry to Simulation mode the simulation menu options and their short cut icons in the tool bar shall become selectable.

- On entry to Simulation mode the project shall be compiled and Console view shall display the errors and warnings.

Behavior:

- In Simulation mode Stop_Simulation shall cause a transition to Edit::Design::Component mode.
- In Simulation mode component state shall be changed by right clicking on them.

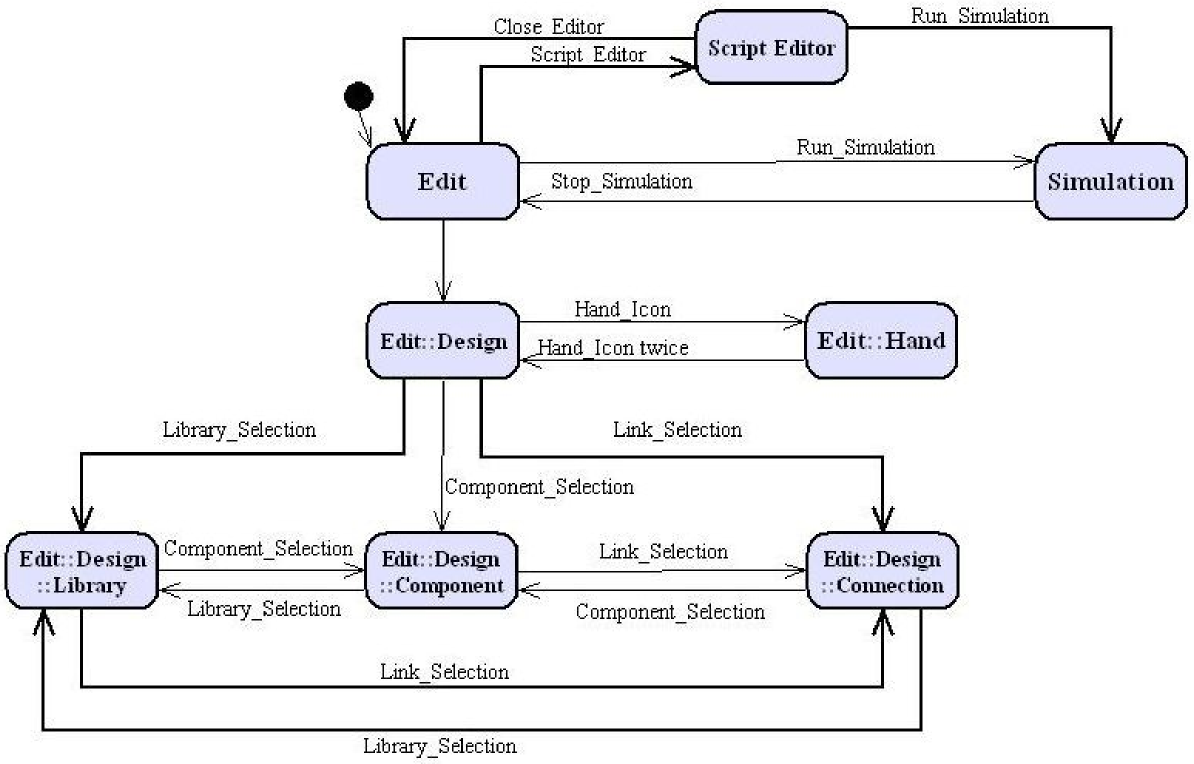
7.2.8 Script Editor

Script Editor mode is where the user is able to build or edit script files by using the text editor. In edit mode, Bellatrix tool presents a standard editor allowing the user to add script files from the macro menu, write source code, find and replace text, etc...

Behavior:

- In Script Editor mode Close_Editor shall cause a transition to Edit::Design mode.
- In Script Editor mode Run_Simulation shall cause a transition to Simulation mode.

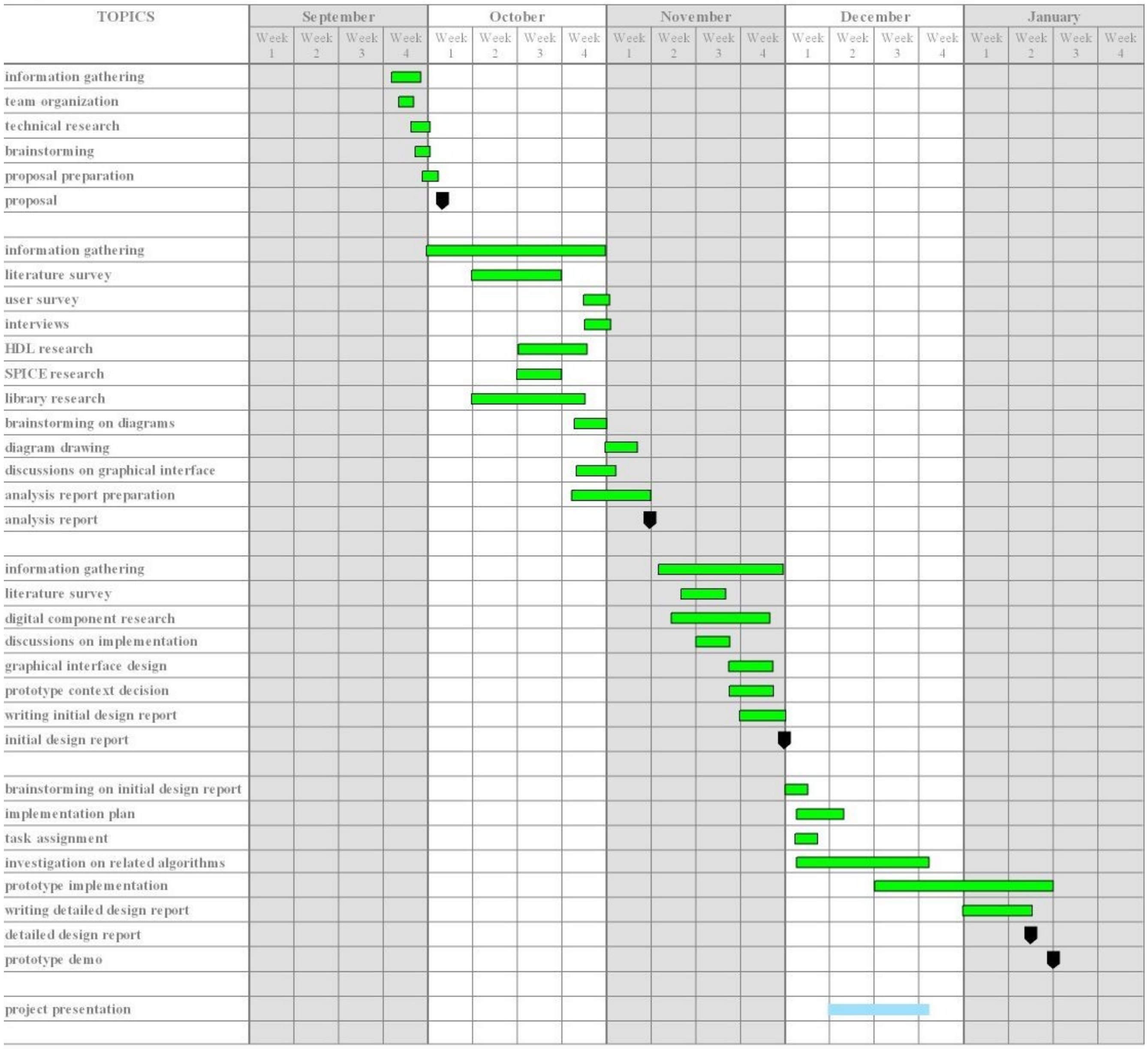
7.3 State Chart



8 Project Plan

information gathering	25.09.2005 – 28.09.2005
team organization	26.09.2005 – 27.09.2005
technical research	27.09.2005 – 30.09.2005
brainstorming	28.09.2005 – 30.09.2005
proposal preparation	30.09.2005 – 01.10.2005
proposal	Milestone
information gathering	01.10.2005 – 01.11.2005
literature survey	07.10.2005 – 21.10.2005
user survey	28.10.2005 – 01.11.2005
interviews	28.10.2005 – 01.11.2005
HDL research	14.10.2005 – 28.10.2005
SPICE research	14.10.2005 – 21.10.2005
library research	07.10.2005 – 28.10.2005
brainstorming on diagrams	24.10.2005 – 30.10.2005
diagram drawing	31.10.2005 – 05.11.2005
discussions on graphical interface	25.10.2005 – 01.11.2005
analysis report preparation	24.10.2005 – 07.11.2005
analysis report	Milestone
information gathering	08.11.2005 – 30.11.2005
literature survey	12.11.2005 – 19.11.2005
digital component research	11.11.2005 – 26.11.2005
discussions on implementation	15.11.2005 – 20.11.2005
graphical interface design	20.11.2005 – 28.11.2005
prototype context decision	20.11.2005 – 28.11.2005
writing initial design report	23.11.2005 – 30.11.2005
initial design report	Milestone
brainstorming on initial design report	01.12.2005 – 04.12.2005
implementation plan	02.12.2005 – 09.12.2005
task assignment	02.12.2005 – 05.12.2005
investigation on related algorithms	02.12.2005 – 23.12.2005
prototype implementation	15.12.2005 – 15.01.2006
writing detailed design report	01.01.2006 – 10.01.2006
detailed design report	Milestone
prototype demo	Milestone
project presentation	07.12.2005 – 23.12.2005 (*to be determined*)
implementation	*to be determined*
testing	*to be determined*
maintenance	*to be determined*
product release	Milestone

Project Time Chart



9 References

[1] The Java Language and Java Environment, http://java.sun.com
[2] Eclipse SDK, www.eclipse.org
[3] Python Scripting Language, www.python.org
[4] Jython, Java port of Python language, www.jython.org
[5] The Hamburg VHDL Archive, http://tech-www.informatik.uni-hamburg.de/vhdl/
[6] The Spice Home Page, http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/
[7] 5Spice Home Page, www.5spice.com
[8] METU Ceng232 Course Page, Diglog, http://www.ceng.metu.edu.tr/courses/ceng232/
[9] Deeds Home Page, http://www.esng.dibe.unige.it/netpro/Deeds/Index.htm
[10] Ksimus Home Page, www.ksimus.berilos.de
[11] Xilinx Home Page, www.xilinx.com
[12] MicroDev Home Page, http://perso.wanadoo.fr/udev
[13] Digital Works 95 Home Page, http://www.spsu.edu/cs/faculty/bbrown/circuits/howto.html
[14] OpenOffice 2.0 Home Page, www.openoffice.org
[15] SmartDraw Home Page, www.smartdraw.com