CENG 491

DIGIMOD
INITIAL DESIGN REPORT

GROUP NAME: **MANAS YAZILIM**

GROUP MEMBERS: **Gökhan ÇAPAR**

Hüseyin B. AYYILDIZ

A. Serhat DEMİR

Seltchouk AHMED

1. INTRODUCTION

1.1 Problem Definition and Project Scope

1.1.1_Problem Definition

Resulting product of our seven month development process is a digital logic simulator which we named DIGIMOD. DIGIMOD let users to draw and plot logic circuits, open or save files, store current circuit into library, import and export LGF files, zoom in and zoom out, use its own script...etc. With its user friendly GUI the purpose of DIGIMOD is to fill in digital logic simulator family with its fascinating features which are all free.

Detailed aspects, objectives, goals, functionality of DIGIMOD are stated in the following sections.

1.1.2_Goals and Objectives

We aim to build full functional digital logic simulator which will make developing and working in logic circuits much easy then ever. We will handle this with our script which brings several fascinating incomes. Overall features of DIGIMOD's script can be found in following sections.

We want DIGIMOD to have user friendly GUI. Most of free digital logic simulators are bad designed in visual way, and the ones who have fascinating GUI's are need to be paid.

Our goal is to submit a free, graphically well designed digital logic simulator to users especially for academic purposes.

Another goal for DIGIMOD is that it will be a totally safe and stable product. As we stated above we hope DIGIMOD will also be used for academic purposes it is important for the product to be stable and safe. A failure during the logic circuit development process would cause failure of the user and DIGIMOD would be an unreliable product for users which actually we don't want to face.

We believe that this system will increase the number of people using this kind of tools, because as well as providing a practical way of developing circuits, our system contains a multifunctional script which eases life.

1.1.3_Statement of Scope

DIGIMOD is a powerful multi platform program that can be used while working on digital logic circuits.

A user will be able to,

- develop circuits with this tool.
- save current file.
- open an existing file.
- send gates and circuit items in to library.
- retrieve gates and circuit items from library.
- edit circuits by cut/copy/paste/move features.
- use zoom in and zoom out on circuit panel in order to improve visual needs.
- print desired circuits.
- import/export LFG files.
- derive appropriate circuit with a logic function using script.
- derive output of a circuit by supplying input using script.

This information is given in order to state the scope of our project and product not to explain how it is handled. We will go into details in further parts of this report.

2. DESIGN CONSTRAINTS

2.1 Constraints

DIGIMOD should be carefully designed in order to work efficiently and satisfy requirements stated before. On the other hand there are some design constraints which we have to consider while designing our program.

Since DIGIMOD will be platform independent we have several restrictions for choosing development tools. We may encounter several problems because of this constraint. We will handle this problem using JAVA tools.

DIGIMOD will use LGF files during its operations which forces us to dissolve LGF file format. Since it is worthwhile to use DIGLOG's library this constraint becomes very important.

2.2 Limitations

- Time limitation. Since we have six months left we have to prepare an adequate schedule. We also have to comply with this schedule in order to success.
- Hardware limitation. In order to give enough sources to development tools and DIGIMOD, we will use at least 1000 MHz CPU, 256 Mb RAM, 40 GB hard drive.
- Portability. As stated before DIGIMOD will be platform independent program. It can be used in every operating system safely.
- Programming language. Our final decision of programming language is JAVA. At the time of our analysis part we were thinking to use .NET as our development platform. But in order not to separate our development tools we recede from using .NET since we will have to use a few tools for scripts, simulators and some graphic needs. We will handle all this within JAVA platform. But there were other choice using C++ but we give up that idea too with same reason. We are planning to integrate simulator in to our platform and other facilities like graphic libraries and script languages will be handled all in to same interface

3. ARCHITECTURAL DESIGN

3.1 Scripting

We will supply an interpreter to support scripting. Except from drawing and analyzing circuits, every feature of the program can be used via interpreter. One of the useful usages is; it supports giving inputs to the circuit and getting outputs without opening main program. Interpreter works independently from the main program but it will also be able to work while the main program is running.

In our interpreter, first a user loads a file. This loaded file can be an existing circuit file, a diglog file or a text file containing a Boolean function. After loading a file, user may give inputs and get output without opening the main program. He/she may want to convert the loaded file into a diglog file or convert a loaded diglog file into our circuit file format.

Another function that our interpreter supports is printing. Again without opening the main program user can get the printout of the loaded file either directly from the printer or print the circuit into a file. Printed file types are postscript file and portable document file formats.

When our interpreter is run with the program, user will be able to send inputs through interpreter during circuit simulation. This feature will be very helpful during simulation running and analyzing process.

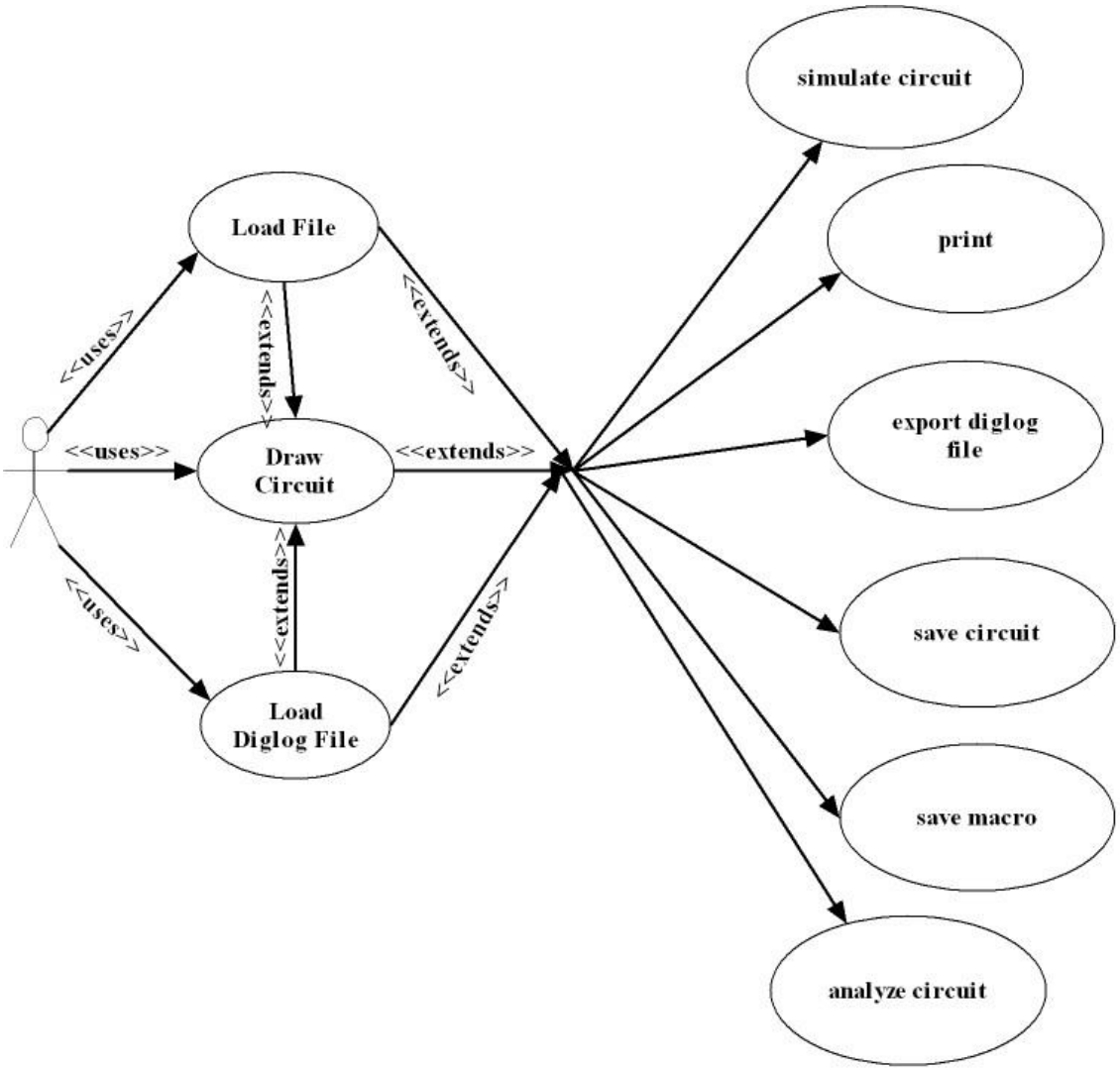
3.2 Main Program

All features supported by scripting available with a user interface that is not complex and self-explanatory. In addition, drawing or editing circuit files can only be done in main program.

4. MODELLING

4.1. Unified Modeling Language Design

4.1.1_Program Use Case



Program Use Case Diagram

Load File

Objective	*To allow user to load a previously saved circuit file.*
Precondition	-
Main Flow	1. *The user interacts with the main window of the program.* 2. *User loads a file by using load button.* 3. *After pressing "open" button standard browse window appears.* 4. *User finds desired file and presses "open" button to finish operation.*

Draw Circuit

Objective	*To allow user to draw circuit.*
Precondition	-
Main Flow	1. *User can draw a circuit using component library, basic circuit elements or previously saved macros.* 2. *User chooses which circuit element to be used then presses related button and finally places it wherever he likes.*

Load Diglog File

Objective	*To allow user to load a previously saved .LGF file.*
Precondition	-
Main Flow	1. *The user interacts with the main window of the program.* 2. *User loads a .LGF file by using load button.* 3. *After pressing "open" button standard browse window appears.* 4. *User finds desired file and presses" open" button to finish operation.*

Simulate Circuit

Objective	*To allow user to control circuit by simulation*
Precondition	*A circuit file must be opened or drawn.*
Main Flow	1. *The user interacts with the main window of the program.* 2. *User presses "simulate" button. So logic high and logic low wires and virtual leds are shown on screen in different colors.*

Print

Objective	*To allow user to get a hard copy or printable soft copy of the current circuit.*
Precondition	*A circuit must be opened or drawn.*
Main Flow	1. *The user interacts with the main window of the program.* 2. *User presses "print" button.* 3. *After pressing print button a selection window appears.* 4. *User can select directly printing, printing to a .PS file or .PDF file.*

Export Diglog File

Objective	*To allow user to create a .LGF file from current circuit.*
Precondition	*A circuit must be opened or drawn.*
Main Flow	1. *The user interacts with the main window of the program.* 2. *User presses "save as" button.* 3. *After pressing button a window appears and asks the name and location of new .LGF file.*

Save Circuit

Objective	*To allow user to save current circuit in a file.*
Precondition	*A circuit must be opened or drawn.*
Main Flow	1. *The user interacts with the main window of the program.* 2. *User presses "save" button.* 3. *After pressing "save" button a window appears and asks the name and location of new file.*

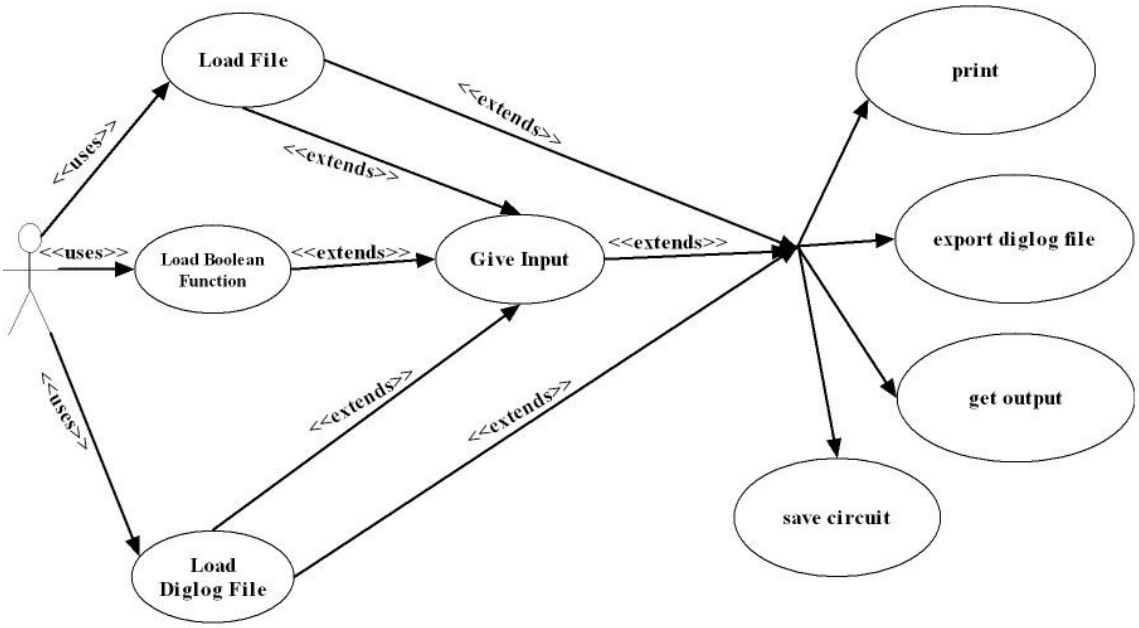
Save Macro

Objective	*To allow user to create macro from current circuit.*
Precondition	*A circuit must be opened or drawn.*
Main Flow	1. *The user interacts with the main window of the program.* 2. *User presses "Save As Macro" button.* 3. *After pressing the button a window appears and asks to specify input and output names and tha name of the macro.*

Analyze Circuit

Objective	*To allow user to analyze a circuit.*
Precondition	*A circuit file must be opened or drawn.*
Main Flow	1. *The user interacts with the main window of the program.* 2. *User presses "analyze" button.* 3. *After pressing button a window appears and user drags and drops which elements that he wants to analyze.* 4. *If user wants to analyze a wire, he can put an output led element and use it.*

4.1.2_Scripting Use Case



Scripting Use Case Diagram

Load file

.

Objective	*To allow user to load a previously saved circuit file.*
Precondition	*File that will be loaded must be in the program directory. Otherwise full path of the file must be entered.*
Main Flow	5. *The user interacts with the interpreter* 6. *User loads a file by typing <load fileName>* 7. *User is informed about success of the load operation.*

Load .LGF file

Objective	*To allow user to load a previously saved .LGF file.*
Precondition	*File that will be loaded must be in the program directory. Otherwise full path of the file must be entered.*
Main Flow	1. *The user interacts with the interpreter* 2. *User loads a .LGF file by typing <load fileName>* 3. *User informed about success of the load operation.*

Load Boolean Function

Objective	*To allow user to load a previously saved Boolean function from a text file.*
Precondition	*File that will be loaded must be in the program directory. Otherwise full path of the file must be entered.*
Main Flow	1. *The user interacts with the interpreter* 2. *User loads a file by typing <load fileName>* 3. *User informed about success of the load operation.*

Give input

Objective	*To allow user give inputs to loaded circuit.*
Precondition	*A circuit file must be loaded successfully.*
Main Flow	1. *The user interacts with the interpreter* 2. *User enters inputs according to loaded circuit.* 3. *If given inputs does not match user is warned.* 4. *After entering proper inputs, according outputs are printed to the screen.*

Printing

Objective	*To allow user printing out a loaded circuit file.*
Precondition	*A circuit file must be loaded successfully.*
Main Flow	1. *The user interacts with the interpreter* 2. *User prints a file by typing <print>* 3. *If user wants to print to file (.ps) or (.pdf), he uses <print –f> or <print –p> respectively and file name to be saved should be entered afterwards.* 4. *If user does not specify a file name file automatically saved with then name "out.ps" or "out.pdf".* 5. *User informed about success of the printing operation.*

Export Diglog File

Objective	*To allow user to create a .LGF file from current circuit.*
Precondition	*A circuit file must be loaded successfully.*
Main Flow	1. *The user interacts with the interpreter* 2. *User prints a file by typing <export fileName>* 3. *User informed about success of the exporting operation.*

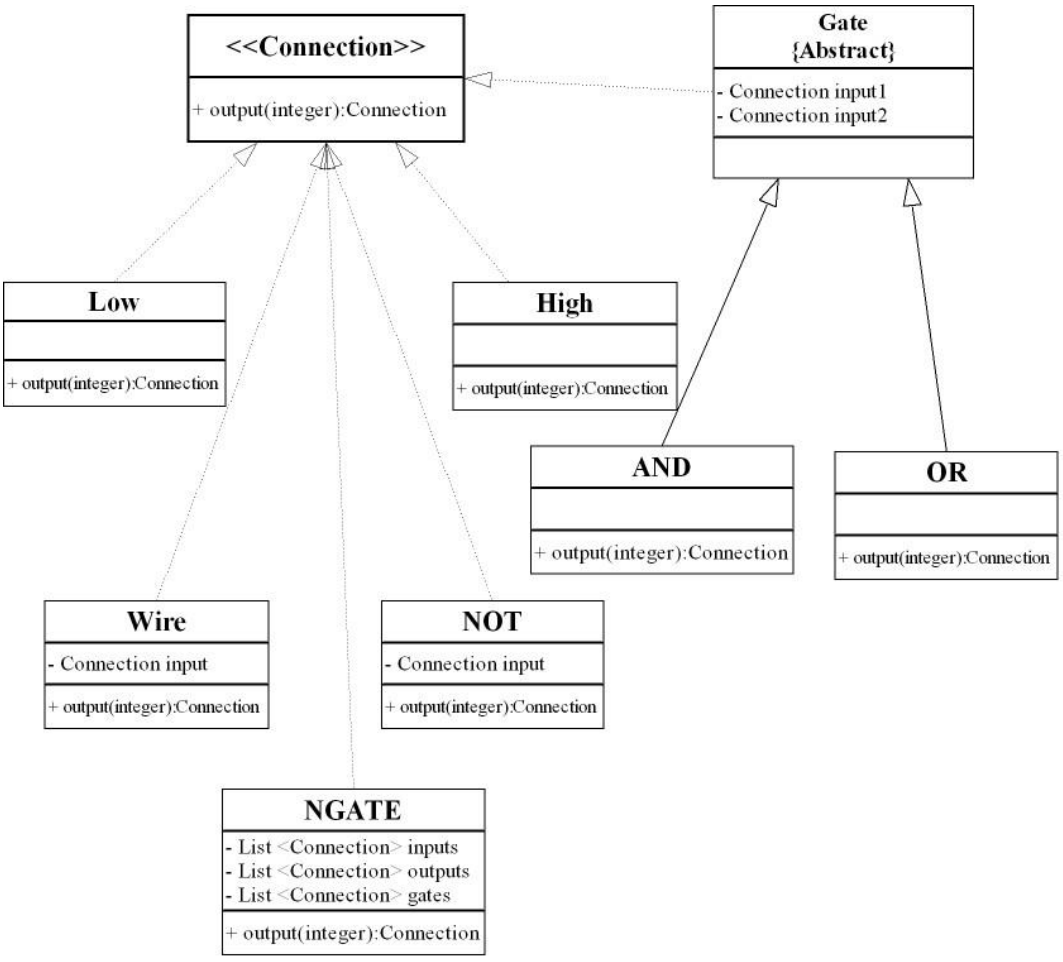
Get Output

Objective	*To allow user to get output from current circuit.*
Precondition	*A circuit file must be loaded successfully.*
Main Flow	*1. User can get output of the current circuit as described in Give Input use case.*

Save Circuit

Objective	*To allow user to create circuit file from current circuit.*
Precondition	*A circuit file must be loaded or generated from Boolean function file successfully.*
Main Flow	*1. The user interacts with the interpreter* *2. User saves a file by typing <save fileName>* *3. User informed about success of the saving operation.*

4.1.3_Class Diagram

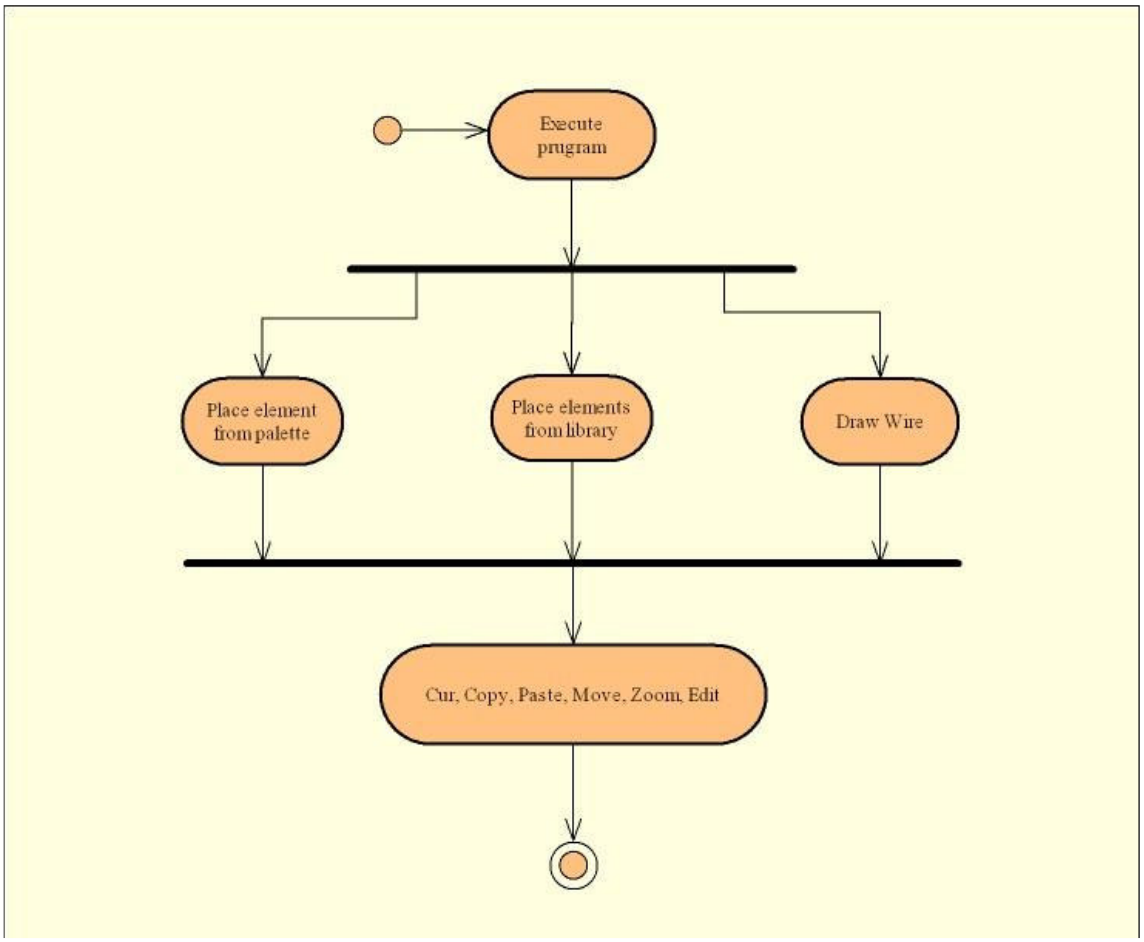


In our background class design, we have an interface class named Connection. It represents any connection point that can have a logical value (high or low). Low and High classes extends Connection interface and returns logical value of related connection point. Wire and NOT classes are also inherited from Connection class, because they give logical values of related connection points.

Gate is an abstract class which implements Connection Interface class. input1 and input2 attributes represents two connection objects that keeps inputs of AND & OR gates. The output(int) function returns output of related gate. AND & OR gates are inherited from that class. We will form all other circuit elements using combinations of AND, OR and NOT gates. These elements can have multiple inputs and outputs. Any input is a connection coming from another element or wire; and any output is a connection that comes from a gate or

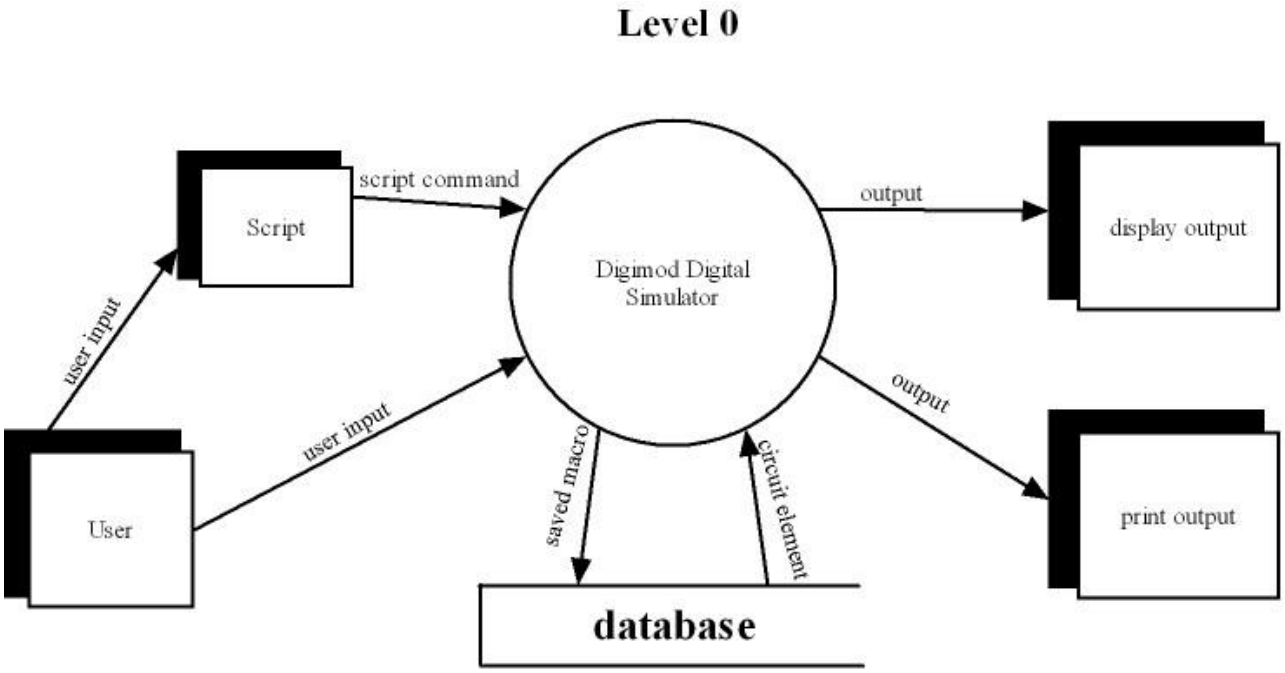
another element inside the element. So, we will use NGATE class to keep complex circuit elements. Inputs and outputs of an element will be kept as two connection object lists inside the NGATE class and another list of connection objects is kept in order to store gates that construct the element.

4.1.4_Drawing Activity Diagram

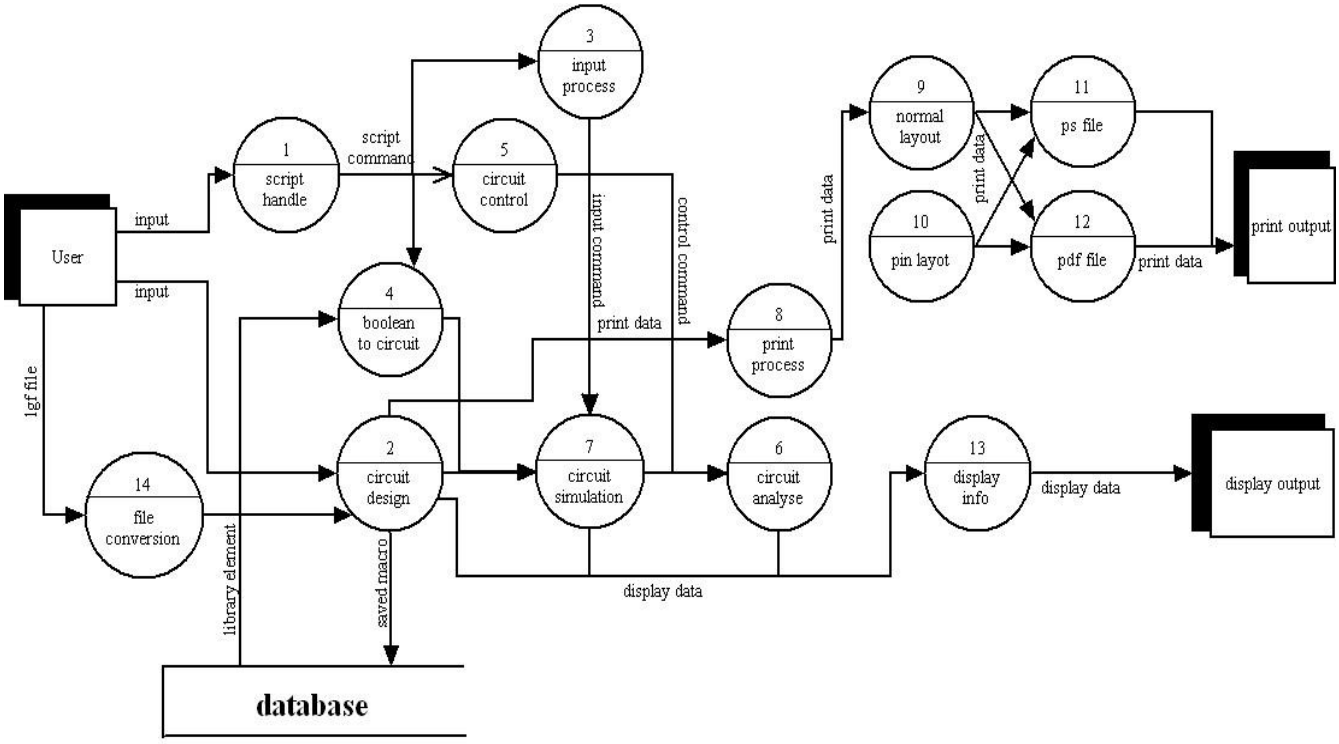


4.2 Data Flow Diagrams

4.2.1_Level 0 DFD



4.2.2_Level 1 DFD



4.2.3_ DFD Description

PSPECS – The Process Specifications

PSPEC: circuit design

Since our program is a circuit simulator. It will also allow users to design their circuits by hand, using the built in elements in the library. After a new circuit is designed, it can be added as a new element to the library so that it will avoid complexity of bigger designs and it will be available for future use.

PSPEC: input process

This is an important feature for our program. Program will accept inputs with the help of the script and users will access outputs without running the program. This will save a lot of time during processing of large number of files.

PSPEC: circuit control

During simulation and analysis of a circuit, users may want to change inputs or state of the circuit. This will be also handled in our program so that users can modify and see circuit behavior while the program is running.

PSPEC: Boolean to circuit

Our program will be able to read Boolean functions and turn them into logic circuits.

PSPEC: circuit simulation

After designing or importing a circuit, users may simulate their designs to see if it works correctly.

PSPEC: circuit analysis

This feature will help users to find their errors and also see the behavior of the circuit. We will monitor what is going on during simulation using graphs.

PSPEC: print process

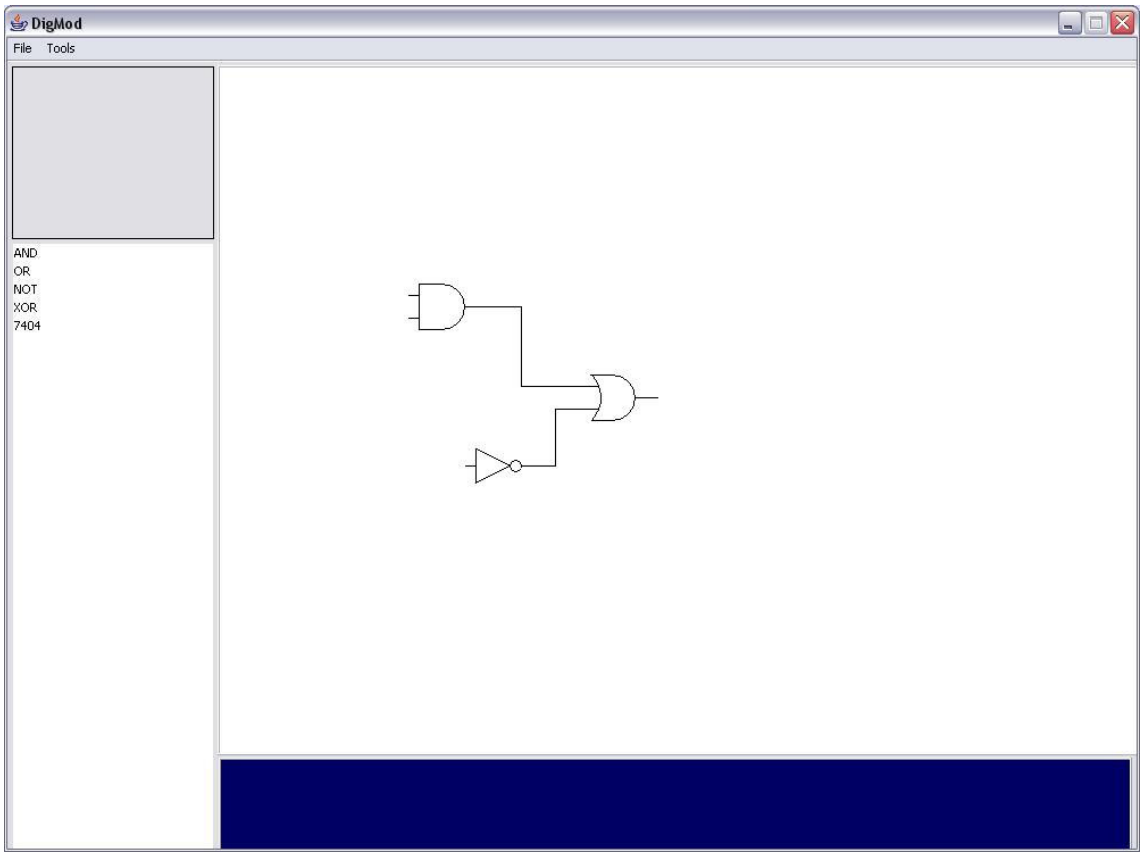
We will have two main types of print support. First is directly getting printout from the printer and the second one is printing to a file. Users can have both PS and PDF file formats available for printouts.

PSPEC: file conversion

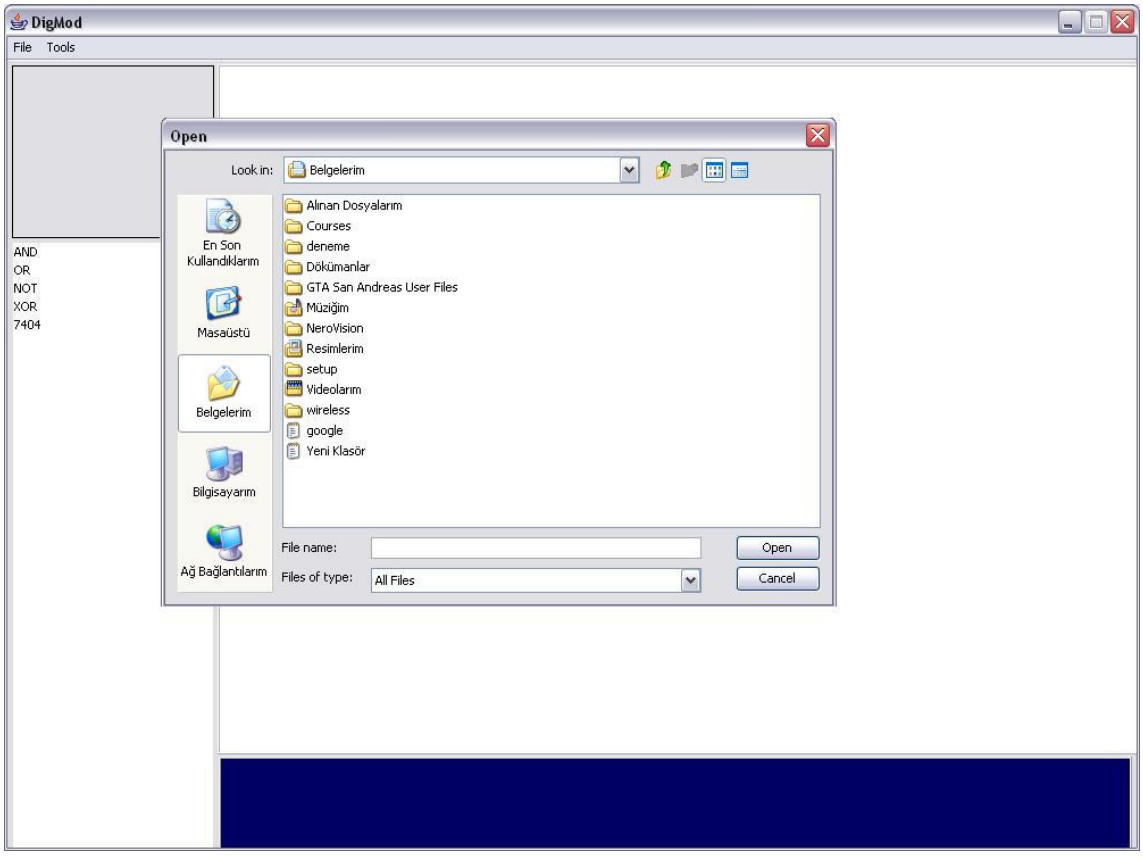
Our program will support also diglog file format. It will import LGF files and process on their circuits.

5. GUI DESIGN

Our Graphical User Interface is simple but functional and self-explanatory. When user executes the program an empty screen arises. Our main program will look like as follows in the beginning.



There is a palette on the left. User chooses elements from there and places in the drawing area. A component for analyzing circuits is placed below the drawing area. There will be a tool bar above the screen for operations as save, load…etc. A screenshot is shown below when an open file dialog is opened.


DigMod
File Tools
AND
OR
NOT
XOR
7404
Open
Look in: Belgelerim
En Son Kullandıklarım
Masaüstü
Belgelerim
Bilgisayarım
Ağ Bağlantılarım
Alınan Dosyalarım
Courses
deneme
Dökümanlar
GTA San Andreas User Files
Müziğim
NeroVision
Resimlerim
setup
Videolarım
wireless
google
Yeni Klasör
File name:
Files of type: All Files
Open
Cancel

6. HELP DESIGN

Help menu is very helpful for lots of software products as it is in DIGIMOD. Since we will create a new program for the digital logic world, users who are used to draw on current simulators may not become easily accustom to our product. Also for new users who are also new to logic methodology will probably need a help part. Besides, our script functionality should be explained briefly in help menu in order to make this feature easily adaptable. On these grounds we can conclude that help menu is an essential part of our project.

We have considered a number of design issues to help the user to handle the system easily which are explained below:

- Help menu must be available at all times during system interaction. A user may need the help menu at the beginning or in the middle of an application. For all this situations, whenever it is needed one can read help file. If help menu is clicked during any application, help page is opened on the same level and after user finish with help page, he/she will click “back” button to return to the current application without loosing any data.

- As partially explained above, help document will be available from help menu at the top of product, also there may be recently and frequently used buttons where one can reach help menu too. Finally there will be a keyboard shortcut in order not to be totally dependent to graphical user interface and to make a fast access.

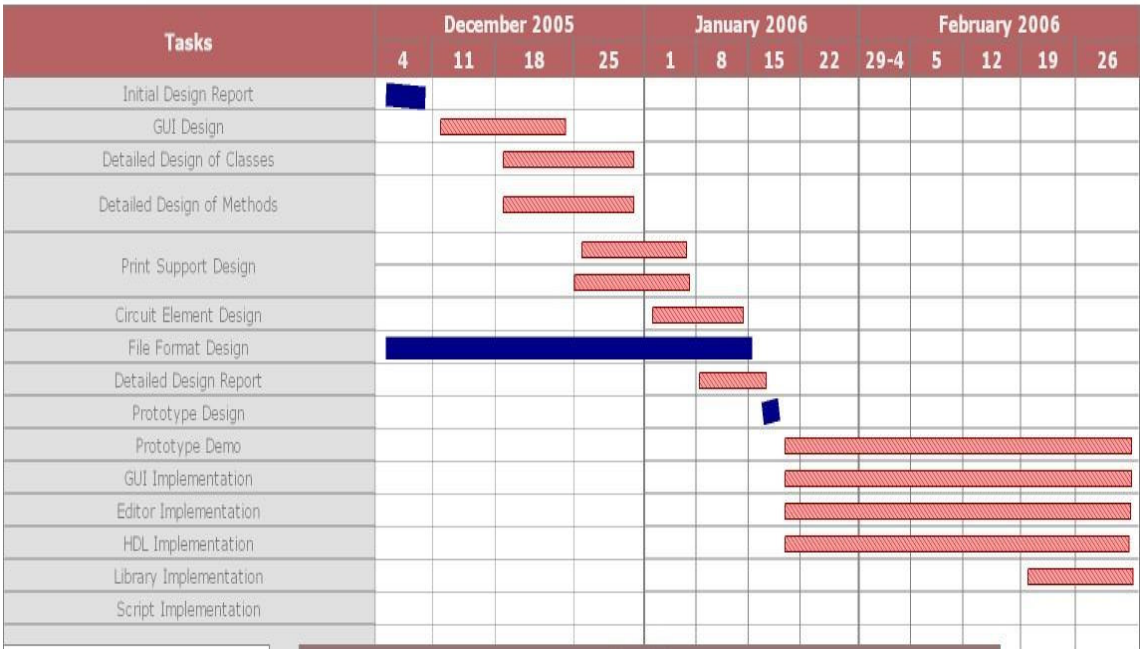
- All functionality and script features will be explained briefly in help menu. We want our help section to be a helpful one. It will comply with all necessary formalities.

- We will structure help section as one can reach what he/she seeks as fastest as possible. In order to provide this functionality we will create layered architecture of information to ensure increasing detail as user goes further.

7. SCHEDULING

7.1 Gantt Chart

GANTT CHART



8. CONCLUSION

Aim of this initial design report is to form a way to create methods of our design which will show the way of our implementation. Data given in this report such as use cases, class diagrams, Gantt chart will be our guide to implement DIGIMOD properly in time. This report is also beneficial for letting us to create a milestone for our project and preparing a pre-design report before final design report that leads perfection in design. As a result this report is an essential part of our project.