

Software Design Specification

Team Groundhog Spring 2009

1. Introduction

1.1 Purpose of this document

This is the Software Design Specification for Team Groundhog's GDS II Viewer application. This document will outline the software design and specification of our application in addition to system architecture, system components, and software requirements as agreed upon by the customer and the project team.

Our GDS Viewer is a .Net based product that is developed in C# with OpenGL Libraries.

1.2 Scope of the development project

The application is a Windows based .Net implementation with OpenGL. Our main UI will be developed in C#, which is a .Net language. This application will allow viewing of GDS II files, along with printing on an Epson Large Format plotter.

1.3 Definitions, acronyms, and abbreviations

- Alpha Blending - Convex combination of two colors allowing for transparency effects in computer graphics.
- GDS II: database file format which is the de facto industry standard for data exchange of integrated circuit or IC layout artwork
- OpenGL – The industry's most widely used, supported and best documented 2D/3D graphics API
- SRS - Software Requirements Specification
- UI - User Interface
- VLSI – Very Large Scale Integration

1.4 References

http://en.wikipedia.org/wiki/Alpha_compositing

<http://en.wikipedia.org/wiki/VLSI>

<http://en.wikipedia.org/wiki/GDSII>

<http://www.opengl.org/documentation/>

1.5 Overview of document

This document will contain specifics about the design of our software. It is a high level explanation of how the application will function, and which components will be developed to

-

help it do so. This document will also serve to explain the technologies used to develop the application, as well as the user's interaction with the application.

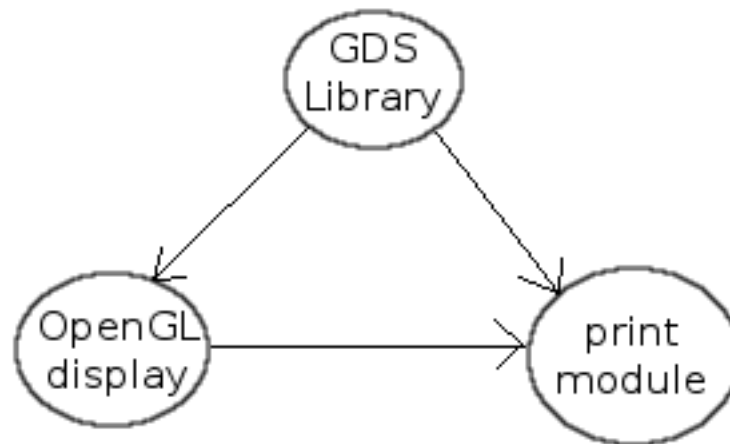
2. System architecture description

2.1 Overview of modules / components

The main components of our system will be the OpenGL VLSI display component which will display the rendered VLSI circuit, the GDS library which parses and builds out the GDS object and the print module which will print the rendered VLSI circuit.

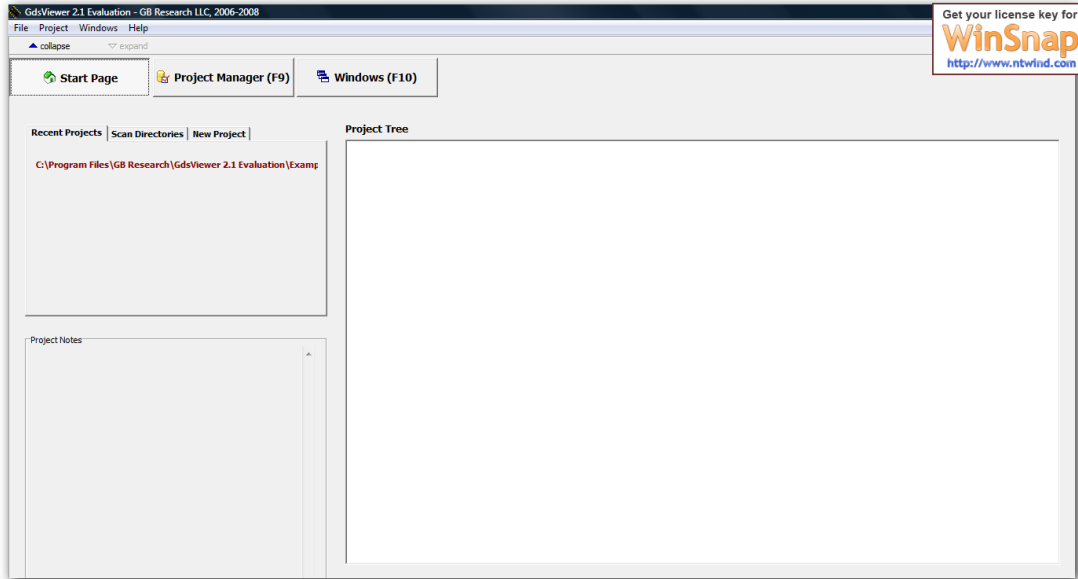
2.2 Structure and relationships

- The OpenGL display depends upon the GDS library.
- The print module depends upon the GDS library

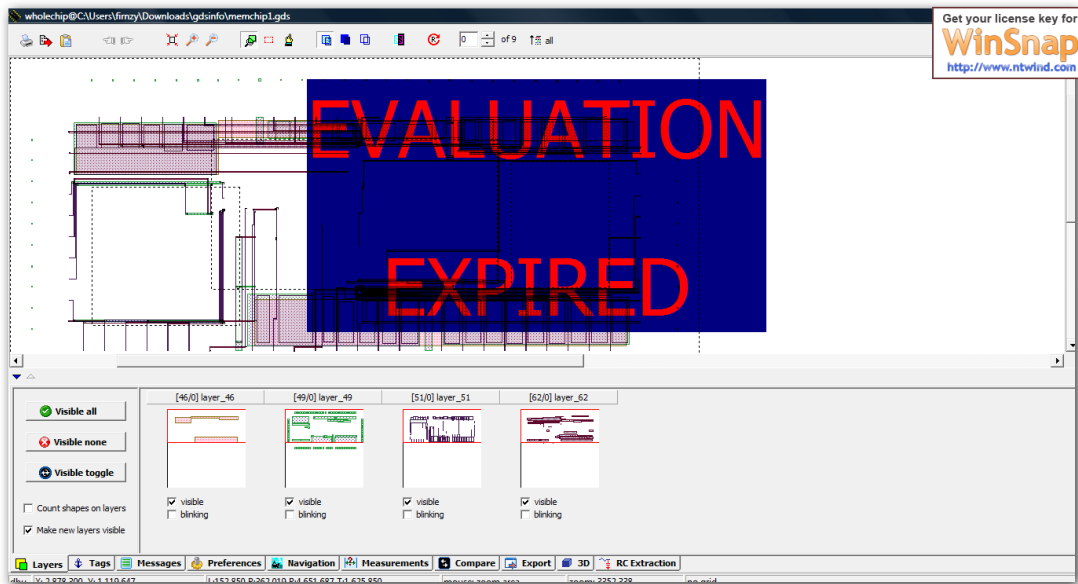


2.3 User interface issues














Our project is focused on displaying VLSI circuits more realistically (alpha blending of layers as opposed to limited stipples). Therefore we will focus on creating a robust UI with an easy to use limited user functionality. This will primarily be a display tool rather than an editor.



Main Window from GDSViewer2.1, which was recommended by sponsor for reference.



Chip view from program recommended by sponsor for reference.

index	name	z1 (μm)	z2 (μm)	color
25/0	layer_25	0	0	
26/0	layer_26	0	0	
42/0	layer_42	0	0	
43/0	layer_43	0	0	
44/0	layer_44	0	0	
45/0	layer_45	0	0	
46/0	layer_46	0	0	
49/0	layer_49	0	0	
50/0	layer_50	0	0	
51/0	layer_51	0	0	
52/0	layer_52	0	0	
61/0	layer_61	0	0	
62/0	layer_62	0	0	

Layer color selector taken from recommended application.

3.0 Detailed description of components

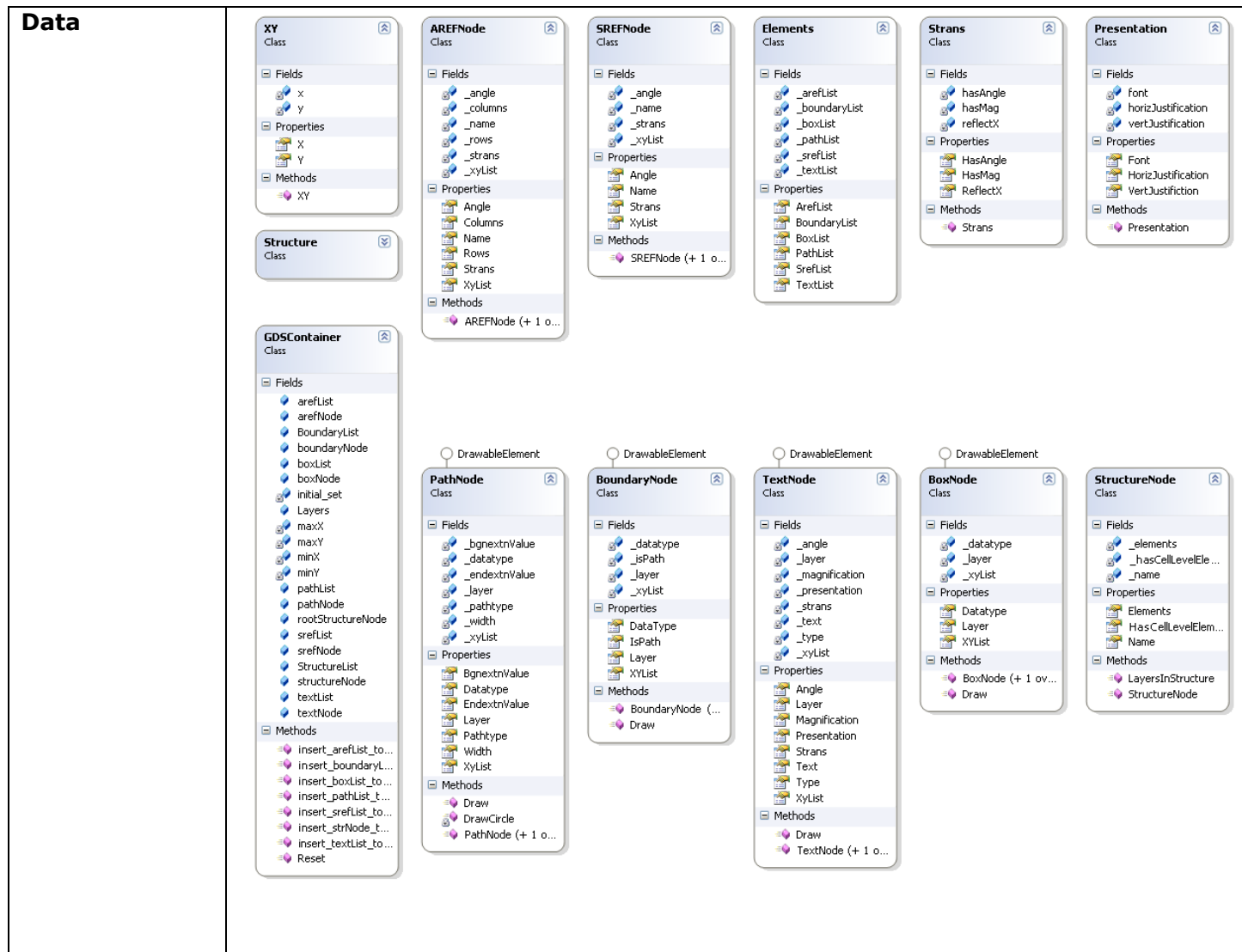
3.1 Component Template Description

Identification	The unique name for the component and the location of the component in the system.
Type	A module, a subprogram, a data file, a control procedure, a class, etc.
Purpose	Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS, but are implied or adjunct to formally stated SDS requirements.
Function	What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified.
Dependencies	How the component's function and performance relate to other components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components.
Interfaces	Detailed descriptions of all external and internal interfaces as well as of any mechanisms for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here.
Resources	A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services.
Processing	The full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic.

Data	For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary.
-------------	---

3.2 GDS Parser

Identification	GDSReader.dll
Type	Module
Purpose	This module is responsible for reading in the GDS file and converting it into usable .Net objects. These objects are then transferred to the Drawing Manager for use in the UI.
Function	It will accept a valid GDS file and read through it 2 bytes at a time. Based on the GDS standard, it generates layers, structures, and drawable objects. These objects are stored on the GDSReader for use by the Drawing layer.
Dependencies	This component is directly dependent on the GDS file provided by the end user. The Drawing component is dependent on this module generating .Net objects for use. It will be called after the end user completes an Open File dialog, at the beginning of their session. The objects generated are handed off to the Drawing Manager.
Interfaces	This module will be instantiated by the UI and provides interfaces to pass in the File name. Errors will be raised if the file is not found, or if it is not in the correct format. The objects that result from parsing the file will be handed off to the Drawing manager.
Resources	CPU Execution Time: <1 Minute Memory: <2 GBs I/O Channels: Hard Drive channel
Processing	GDSReader.Load(string FileName) //Loads the file selected by the user GDSReader.Read() //Iterates through the GDS file, reading 2 bytes at a time. //Applies the GDS structure to create relevant .Net Objects GDSReader.GDSContainer //Resulting property that contains the .Net Objects



3.3 User Interface

Identification	User Interface.
Type	Module.
Purpose	Serves to provide the user means to access the data that they would wish to alter and visualize.
Function	Functions as the gateway to the visualization of the given GDS file. Will take an input file that will be parsed by the GDS parser and given to the Drawing Manager to be drawn in a window contained in the interface.
Dependencies	Contains all the needed components in the background to draw a given chip by opening a GDS file. The GDS file is passed through the

	GDS Parser to withdraw the needed components of the file. Once parsed, passed to the Draw Manager, which will draw the objects back to the interface. Once finalized, will pass the drawn objects to a printer driver.
Interfaces	The User Interface has a file dialog menu to allow for file input. A drawing window that the Drawing Manager will output to. A component to hold the separate layer components of the file. A hierarchy selector that allows you to see varying depths of a chip. Color selection pallet for the varying layers.
Resources	The user interface will need memory to run. Ability to save open color pallets.
Processing	Open file dialog. Saving the chip as a file with color. Color selection on the various layers. Varying hierarchy depth selection.
Data	Takes a GDS file as input. Stores the color and descriptions for the varying objects of the GDS file.

3.4 Drawing Manager

Identification	Drawing Manager
Type	Library
Purpose	The drawing manager's job is to use OpenGL to draw objects specified by the GDSII Parser.
Function	After the GDSII Parser gathers the data/objects from a specified file, each object will have a method that makes calls to the Drawing Manager, which is responsible for drawing different layers using a color pallet specified by the UI, and rendering the objects using alpha blending where each layer has a different opacity.
Dependencies	The Drawing Manager depends on the User Interface to specify a color pallet to use. The Drawing Manger also depends on the GDSII Parser to supply objects to be displayed.
Interfaces	The Drawing Manager is a library of Static Method calls that can be called from the GDSII Parser. Depending on what is being drawn, there are different method calls that will be used. The Drawing Manager is not responsible for drawing order that is left to the GDSII Parser method to handle.
Resources	Each object drawn allocates a small amount of memory on the graphics card. This memory is then manipulated by each primitive object during its draw() cycle.
Processing	The Drawing Manager will be responsible for processing all transformations, translations, rotations, alpha blending and object rendering. We also plan on including a process for zooming, resizing and graphic object modification.
Data	There is no local data stored in the Graphics Manager, as all data is stored in either the User Interface or in the GDSII Parser.

3.5 Print Module

Identification	Print Module
Type	Module
Purpose	Visually display the print options. The inputs are mouse clicks and keyboard input and the output is the current print options.
Function	This module allows the user to select the appropriate print options as well as submit the print job to the printer.
Dependencies	This module requires that the GDS file has been rendered by the OpenGL display component.
Interfaces	The print module must interface with the OpenGL display component. The OpenGL display component will make the rendered file available to the print module.
Resources	The print modules requires enough memory and CPU resources to transform the rendered GDS file to the appropriate printable format.
Processing	The print module window is spawned via the main UI and the default print options are displayed. User input is received and the values are updated appropriately. Upon user input to print the file, the rendered GDS file is transformed to the appropriate printable format and sent to the printer.
Data	The print object is stored as a byte array.

4.0 Reuse and relationships to other products

GDSViewer – We are looking at this application for comparison to indicate whether our product is working correctly.

OpenGL/Tao – We are using the Tao library to use OpenGL in order to display the objects in the GDSII file.

Other: We have looked at libraries developed for other systems and other languages to get an idea as to how they parse through GDSII files. We are not using these files directly but are using them as reference if we are ever confused about a certain object.

5.0 Design decisions and tradeoffs

We have decided to use OpenGL rather than WPF because of WPF's large overhead, performance issues, and steep learning curve.

We have chosen to develop the application in C# because the client would like a Windows based application.

Using Windows allows us the ability to use built in drivers and print functions for printing.

-

We have decided to not use a library for GDS parsing because of licensing issues and language barriers.

6.0 Pseudocode for components

```
<?xml version="1.0"?>
<doc>
  <assembly>
```

```

    <name>GlInterface</name>
  </assembly>
  <members>
    <member name="T:GlInterface.Properties.Resources">
      <summary>
        A strongly-typed resource class, for looking up localized
strings, etc.
      </summary>
    </member>
    <member name="P:GlInterface.Properties.Resources.ResourceManager">
      <summary>
        Returns the cached ResourceManager instance used by this class.
      </summary>
    </member>
    <member name="P:GlInterface.Properties.Resources.Culture">
      <summary>
        Overrides the current thread's CurrentUICulture property for
all
        resource lookups using this strongly typed resource class.
      </summary>
    </member>
    <member name="M:GlInterface.GlInterface.Init">
      <summary>
        Initialize the OpenGL engine
      </summary>
      <remarks>
        Call this after Form.InitializeComponent() and
SimpleOpenGLControl.InitializeContexts()
      </remarks>
    </member>
    <member
name="M:GlInterface.GlInterface.Resize(System.Int32,System.Int32)">
      <summary>
        Resize the rendering volume to match the SimpleOpenGLControl
dimensions
      </summary>
      <param name="width">SimpleOpenGLControl.Width</param>
      <param name="height">SimpleOpenGLControl.Height</param>
    </member>
    <member name="M:GlInterface.GlInterface.Paint">
      <summary>
        Paint a simple scene.
      </summary>
      <remarks>
        This may be changed in the future to take a STL collection of GDS
Element objects
-
        grouped/sorted by layer (or a collection of sorted Layer objects)
as an input parameter.
      </remarks>
    </member>
    <member name="M:GlInterface.Program.Main">
      <summary>
        The main entry point for the application.
      </summary>
    </member>
    <member name="F:GlInterface.MainForm.components">

```

```

        <summary>
        Required designer variable.
        </summary>
    </member>
    <member name="M:GInterface.MainForm.Dispose(System.Boolean)">
        <summary>
        Clean up any resources being used.
        </summary>
        <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
    </member>
    <member name="M:GInterface.MainForm.InitializeComponent">
        <summary>
        Required method for Designer support - do not modify
the contents of this method with the code editor.
        </summary>
    </member>
    <member
name="M:GInterface.MainForm.GlWindow_Resize(System.Object,System.EventArgs)"
>
        <summary>
        OpenGL resize event
        </summary>
        <remarks>
        Tie this to the resize event on the splitContainer element which
holds the SimpleOpenGLControl
        </remarks>
        <param name="sender"></param>
        <param name="e"></param>
    </member>
    <member
name="M:GInterface.MainForm.GlWindow_Paint(System.Object,System.Windows.Forms.PaintEventArgs)">
        <summary>
        OpenGL paint event
        </summary>
        <remarks>
        Tie this to the paint event on the SimpleOpenGLControl
        </remarks>
        <param name="sender"></param>
        <param name="e"></param>
    </member>
</members>
</doc>

```

```

public class GDSII
{

```

```

-
    #region DATA_TYPES
    protected short NO_DATA           = 0x00;
    protected short BIT_ARRAY        = 0x01;
    protected short TWOBYTE_SIGN_INT = 0x02;    //two's complement
    protected short FOURBYTE_SIGN_INT = 0x03;    //two's complement
    protected short FOURBYTE_SIGN_REAL = 0x04;    //Bit0 = sign; Bit1-7
= exponent; bit8-31 = mantissa; value = (mantissa/(2^24)) * (16^(exponent-
64))

```

```

    protected short EIGHTBYTE_SIGN_REAL = 0x05;      //Bit0 = sign; Bit1-7
= exponent; bit8-63 = mantissa; value = (mantissa/(2^56)) * (16^(exponent-
64))
    protected short ASCII_STRING          = 0x06;
    #endregion
    #region RECORD_TYPES
    /// <summary>
    /// If you are uncertain as to what these mean or do, refer to
RecordTypes.txt
    /// </summary>
    protected short HEADER                 = 0x00;
    protected short BNLIB                 = 0x01; //Beginning of Library
includes mod and access dates
    protected short LIBNAME               = 0x02; //Name of library
typically 6 char name 2 char extension
    protected short UNITS                 = 0x03; //size of db in user
units then size of db in meters
    protected short ENDLIB               = 0x04;
    protected short BGNSTR               = 0x05; //Begin structure, also
mod and access dates
    protected short STRNAME               = 0x06; //Name of structure alpha-
numeric characters and $
    protected short ENDSTR                 = 0x07;
    protected short BOUNDARY              = 0x08;
    protected short PATH                  = 0x09;
    protected short SREF                  = 0x0a;
    protected short AREF                  = 0x0b;
    protected short TEXT                  = 0x0c;
    protected short LAYER                 = 0x0d;
    protected short DATATYPE              = 0x0e; //from 0 to 63
    protected short WIDTH                  = 0x0f; //negative # means
absolute
    protected short XY                    = 0x10; //An array of XY
coordinates.
    protected short ENDEL                  = 0x11; //End Element
    protected short SNAME                  = 0x12; //name of referenced
structure
    protected short COLROW                 = 0x13; //columns and rows for
AREF, 2 Byte integers first col second row
    protected short TEXTNODE              = 0x14; //typically not used
    protected short NODE                  = 0x15;
    protected short TEXTTYPE              = 0x16; //limit to 63
    protected short PRESENTATION          = 0x17; //bit array
    protected short SPACING               = 0x18; //discontinued
    protected short STRING                 = 0x19; // up to 512 char
    protected short STRANS                 = 0x1a; //bit array
    protected short MAG                    = 0x1b;
    protected short ANGLE                  = 0x1c;
    protected short UINTEGER              = 0x1d;
-
    protected short USTRING                = 0x1e;
    protected short REFLIBS               = 0x1f;
    protected short FONTS                  = 0x20;
    protected short PATHTYPE              = 0x21;
    protected short GENERATIONS           = 0x22;
    protected short ATTRTABLE             = 0x23;
    protected short STYPTABLE             = 0x24;

```

```

protected short STRTYPE           = 0x25;
protected short ELFLAGS           = 0x26;
protected short ELKEY             = 0x27;
protected short LINKTYPE         = 0x28;
protected short LINKKEYS         = 0x29;
protected short NODETYPE         = 0x2a;
protected short PROPATTR         = 0x2b;
protected short PROPVALUE        = 0x2c;
protected short BOX              = 0x2d;
protected short BOXTYPE          = 0x2e;
protected short PLEX             = 0x2f;
protected short BGNEXTN          = 0x30;
protected short ENDEXTN          = 0x31;
protected short TAPENUM          = 0x32;
protected short TAPECODE         = 0x33;
protected short STRCLASS         = 0x34;
protected short RESERVED         = 0x35;
protected short FORMAT           = 0x36;
protected short MASK             = 0x37;
protected short ENDMASKS         = 0x38;
protected short LIBDIRSIZE       = 0x39;
protected short SRFNAME          = 0x3a;
protected short LIBSECURE        = 0x3b;
#endregion
#region Private Variables
private int myNumOfLayers;
private int myVersion;
private System.Collections.ArrayList myLayers;
#endregion
#region Getters/Setters
//GETTERS and SETTERS
public int numOfLayers
{
    get { return myNumOfLayers; }
    set{myNumOfLayers = value;}
}

public int version
{
    get { return myVersion; }
    set { myVersion = value; }
}
#endregion

public GDSII(String fileName)
{
    FileStream fs = File.OpenRead(fileName);
    BinaryReader input = new BinaryReader(fs);
    myNumOfLayers = 0;
}

public void addLayer(Layer newLayer)
{
    myLayers.Add(newLayer);
    myNumOfLayers++;
}

```

```

}

public Layer getLayer(int layerNum)
{
    return (Layer)myLayers[layerNum];
}

#region PARSE_METHODS
/**
 * getHeader()
 * Parameters: A binary streamer, a Binary File to read
 *
 * Objective: The get header method will be used to parse out the
header of a GDSII formatted file
 * which contains a version number
 * */
private void getHeader()
{
}

/**
 * getBGNLib()
 * Parameters: A binary streamer, a Binary File to read
 *
 * Objective: The get library will be called when bgnLib is reached
which will get the
 * Library key word, access time, modification time and Library Name
 * */
private void getBGNLib()
{
}

/**
 * getUntis()
 * Parameters: A binary streamer, a Binary File to read
 *
 * Objective: the getUnits will be called to store the user units and
the physical units
 * */
private void getUnits()
{
}

/**
 * getStructure()
 * Parameters: A binary streamer, a Binary File to read
 *
 * Objective:
 * */
-
private void getStructure()
{
}

```

```

    /***
     * getXy()
     * Parameters: A binary streamer, a Binary File to read
     *
     * Objective: getXy will get the points to be used to describe
points, rectangles.
     * rectangles are defined in this order: Left, Bottom, Right, Top
     * */
private void getXy()
{
}

    /***
     * getBoundary()
     * Parameters: A binary streamer, a Binary File to read
     *
     * Objective:
     * */
private void getBoundary()
{
}

    /***
     * getPath()
     * Parameters: A binary streamer, a Binary File to read
     *
     * Objective:
     * */
private void getPath()
{
}

    /***
     * getSREF()
     * Parameters: A binary streamer, a Binary File to read
     *
     * Objective:
     * */
private void getSREF()
{
}

    /***
     * getAREF()
     * Parameters: A binary streamer, a Binary File to read
     *
     * Objective:
     * */
private void getAREF()
{
}

    /***
     * getText()

```

```

    * Parameters: A binary streamer, a Binary File to read
    *
    * Objective:
    * */
private void getText()
{

}

/**
 * getNode()
 * Parameters: A binary streamer, a Binary File to read
 *
 * Objective:
 * */
private void getNode()
{
}

/**
 *Other records
 *Libdirsize ::= "LIBDIRSIZE" number
 *Srfname ::= "SRFNAME" string
 *Libsecur ::= "LIBSECUR" {number}+
 *Reflibs ::= "REFLIBS" {string}+
 *Fonts ::= "FONTS" {string}+
 *AttrTable ::= "ATTRTABLE" {string}+
 *Generations ::= "GENERATIONS" number
 *FormatType ::= Format [{Mask}+ EndMask]
 *Format ::= "FORMAT" number
 *Mask ::= "MASK" string
 *StrName ::= string
 *StrClass ::= "STRCLASS"
 *Box ::= "BOX" [Elflags] [Plex] Layer Boxtype Xy
 *EndLib ::= "ENDLIB"
 * */

#endregion
}
}

```

7.0 Appendices

none