***Design Document***

***Digital Circuit***

**Adrian Yankov**

**Quang Minh Pho**

**Denny Damara**

**Lyubomir Atanasov**

**Content**

Content…………………………………………………………………………… 2

Introduction………………………………………………………………………. 3

Class Diagram………….………………………………………………………… 4

Class and Methods……………………….………………………………………. 5

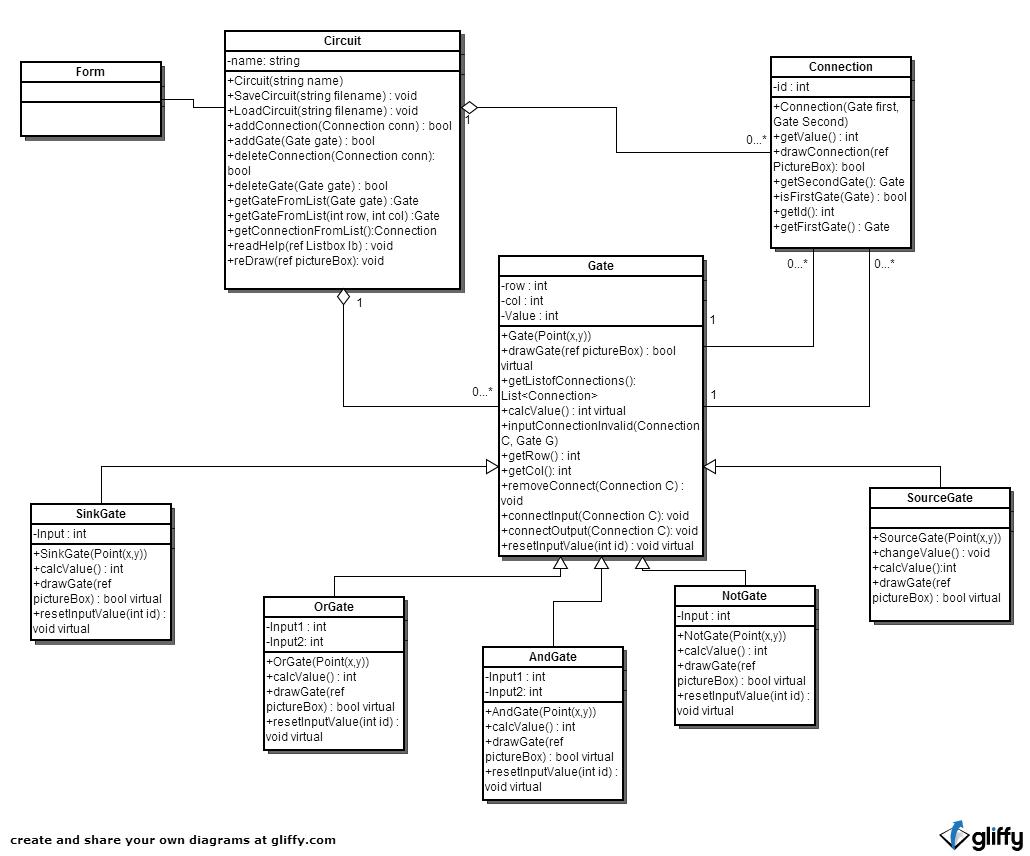
Appendix

Sequence Diagrams……………….………………………………………………. 10

**Introduction:**

The following document contains the class diagram and sequence diagram of the application. Along with the description of the methods contained in the classes.

**Class Diagram**

****

**Class Description**

1. **Circuit:**

This class represents the whole circuit, the master class; it has the list of gates and connections inside.

Attribute:

* -Name
  + The name of the circuit

Methods:

* +Circuit(string name)
  + Constructor of the class,
  + Name is the name of the circuit
* +SaveCircuit(string filename) : void
  + To save the circuit to a file with specified filename
* +LoadCircuit(string filename) : void
  + Loads the circuit from a file with the specified name
* +addConnection(Connection conn) : bool
  + Checks for unallowed connection
  + Returns true if adding connection to conn to the list of connections is successful
* +addGate(Gate gate) : bool
  + Does the checking for overlapping
  + Returns true if adding the gate to the list of gates is succesful
* +deleteConnection(Connection conn): void
  + Deletes connection conn from the circuit and list.
* +deleteGate(Gate G) : void
  + Deletes gate G and the associated connection from list and circuit.
* +getGateFromList(Gate G) :Gate
  + Gets the gate from the list with the same property Gate G.
* +getGateFromList(int row, int col) :Gate
  + Gets the gate from the list with the same row and column number as specified
* +getConnectionFromList(Connection C): Connection
  + Get the selected connection C from the list of connections
* +reDraw(ref PictureBox) : void
  + Redraw the circuit after modified in the picturebox
* +readHelp(ref ListBox lb) : void
  + Get the helptext from help file and write it in the listbox lb

1. **Gate**

Description: Represents a gate.

Attribute:

* -Row : int
  + The row number of the gate
* -Col : int
  + The column number of the gate
* -Value : int
  + The value of the gate
  + Default : -1

Methods:

* +getCoord() : Point
  + Gets the coordination of the calling gate
* +drawGate(ref pictureBox) : void virtual
  + Draws the gate in the pictureBox
* +getListofConnections(): List<Connection>
  + Gets the list of connections connected to gate.
* +calcValue() : int virtual
  + Calculates the value of the gate.
* +inputConnectionInvalid(Connection C)
  + Turns the connection C of the calling gate to invalid
* +Gate(Point (X,Y))
  + Constructor , needs the coordinates of the gate.
  + Translates the coordinates into row and column number
* +getRow() : int
  + Get Row number of the gate
* +getCol() : int
  + Get the column number of the gate
* +removeConnect(Connection C) : void
  + remove the Connection C from the list of connection of the calling gate
* +connectInput(Connection C) : void
  + Checks which input slot is available
  + Connects the connection C as an input for the calling gate
* +connectOutput(Connection C) : void
  + Connects the connection C as an output for the calling gate
* +resetInputValue(int id) : void virtual
  + Resets the input value of the gate depends on the id

1. **Connection**

Description: Represents the collection between gates.

Attribute:

* -FirstGate: Gate
  + The gate which functions as the input for the connection
* -SecondGate: Gate
  + The gate which functions as the output for the connection
* -Id : int
  + The id to identify the connection

Methods:

* +getValue() : int
  + Gets the value of the connection.
* +drawConnection(Gate first, Gate Second): void
  + Draws the connection between two gates.
* +getSecondGate(): Gate
  + Gets the second gate of the connection
* +getFirstGate(): Gate
  + Gets the first gate of the connection
* +isFirstGate(Gate) : bool
  + Is this gate the first of the connection
* +getId() : int
  + get the Id number of the connection

1. **SinkGate**

Description: Inherits from the Gate class. The lamp of the circuit

Attribute:

* Input : int
  + The input for the sink to determine the value of the circuit

Methods:

* CalcValue():int
  + Calculates the value of the gate.
* SinkGate (Point(x,y))
  + Calls the base class constructor and pass the point
* +drawGate(ref pictureBox) : void override
  + Draws the gate in the pictureBox
* +resetInputValue(int id) : void override
  + Resets the input value of the gate depends on the id

1. **SourceGate**

Description: Inherits from the Gate class.

Methods:

* ChangeValue():void
  + Changes the value of the source gate.
* CalcValue():int
  + Returns the value of the source gate
* SourceGate (Point(x,y))
  + Calls the base class constructor and pass the point
* +drawGate(ref pictureBox) : void override
  + Draws the gate in the pictureBox
* +resetInputValue(int id) : void override
  + Resets the input value of the gate depends on the id

1. **OrGate**

Description: Inherits from the Gate class.

Attribute:

* Input1 : int
  + The first input for the or gate to determine the value of the gate
* Input2 : int
  + The second input for the or gate to determine the value of the gate

Methods:

* CalcValue():int
  + Calculates the value of the gate.
* OrGate (Point(x,y))
  + Calls the base class constructor and pass the point
* +drawGate(ref pictureBox) : void override
  + Draws the gate in the pictureBox
* +resetInputValue(int id) : void override
  + Resets the input value of the gate depends on the id

1. **AndGate**

Description: Inherits from the Gate class.

Attribute:

* Input1 : int
  + The first input for the and gate to determine the value of the gate
* Input2 : int
  + The second input for the and gate to determine the value of the gate

Methods:

* CalcValue():int
  + Calculates the value of the gate.
* AndGate (Point(x,y))
  + Calls the base class constructor and pass the point
* +drawGate(ref pictureBox) : void override
  + Draws the gate in the pictureBox
* +resetInputValue(int id) : void override
  + Resets the input value of the gate depends on the id

1. **NotGate**

Description: Inherits from the Gate class.

Attribute:

* Input : int
  + The input for the not gate to determine the value of the gate

Methods:

* CalcValue():int
  + Calculates the value of the gate.
* NotGate (Point(x,y))
  + Calls the base class constructor and pass the point
* +drawGate(ref pictureBox) : void override
  + Draws the gate in the pictureBox
* +resetInputValue(int id) : void override
  + Resets the input value of the gate depends on the id

**Sequence diagram:**

See other document.