# SBROME: A Scalable Optimization and Module Matching Framework for Automated Biosystems Design

**SBROME Version: 0.81**
**Documentation Version: 1.1**
**Date: 12/21/2012**

## 1. OVERVIEW

The **S**ynthetic **B**iology **R**eusable **O**ptimization **Me**thodology (**SBROME**) is a computational platform for the modular, automatic design of synthetic circuits. It is supported by a module and parts database that contains characterized parts and experimentally validated synthetic designs. The SBROME framework first uses graph isomorphism algorithms to match modules to topology and then decompose the resulting topology to sub-circuits that can be efficiently solved through exact or approximate optimization methods. Its methods include graph-theoretic approaches that allow both abstract and specific definitions of parts and modules. As a result, multiple underlying architectures can be considered as possible solutions to any functional or logical relationship, an important characteristic given the mechanistic diversity that underlies biological systems.

**WARNING***: SBROME and its interactive GUI are still work in progress! Although the platform is stable, it has limited functionality and its lacking key features to help HCI and interoperability between the modules. This preliminary version is provided for beta users. We expect an user-friendly version to be released in Spring 2013.*

## 2. SBROME ARCHITECTURE

SBROME take as inputs an abstract circuit schematic, a table with the desired Input/Output values and other optional user constrains. After execution, SBROME returns a fully defined circuit where parts have been selected so that the behavior of the whole circuit satisfies any user-defined constraints and it best approximates the desired I/O characteristics. Figure 1 illustrates the proposed divide-and-conquer approach. In the first step of the SBROME platform, a module library with pre-constructed designs is queried to find possible partial or full matches to the given abstract circuit topology, which has been specified by the user. To do so, both module and abstract circuit networks have to be transformed in an equivalent graph representation that can support the complex relationships and diverse dependencies that are encoded within a synthetic circuit diagram. After we match existing modules to the transformed network, we decompose the circuit into sub-circuits of relatively small size (4-6 components) by applying graph partitioning algorithms that minimize interconnectivity among circuit partitions. The link values are subsequently quantized, which reduces the problem dimensionality and allows for exact methods to be applied efficiently in order to find the part combination that minimizes the difference between the desired and actual gene expression values. Finally, the computationally-inferred sub-circuits are experimentally characterized and then integrated to comprise the final construct. All newly characterized circuits are deposited in the module database for future reuse. More about SBROME can be found in [1]

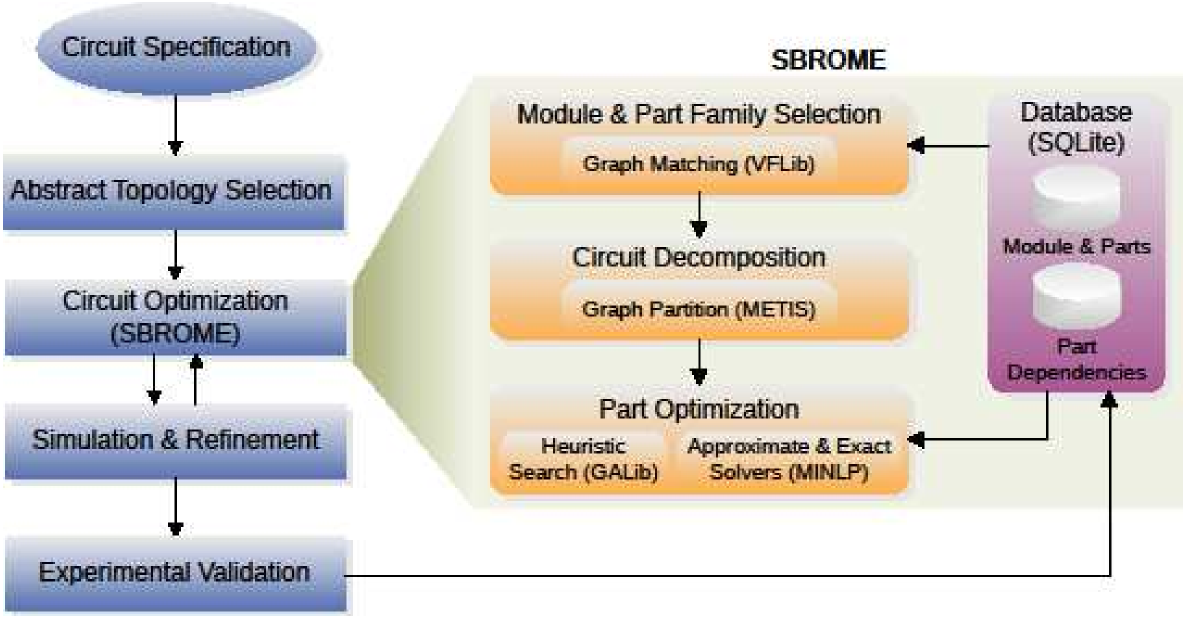
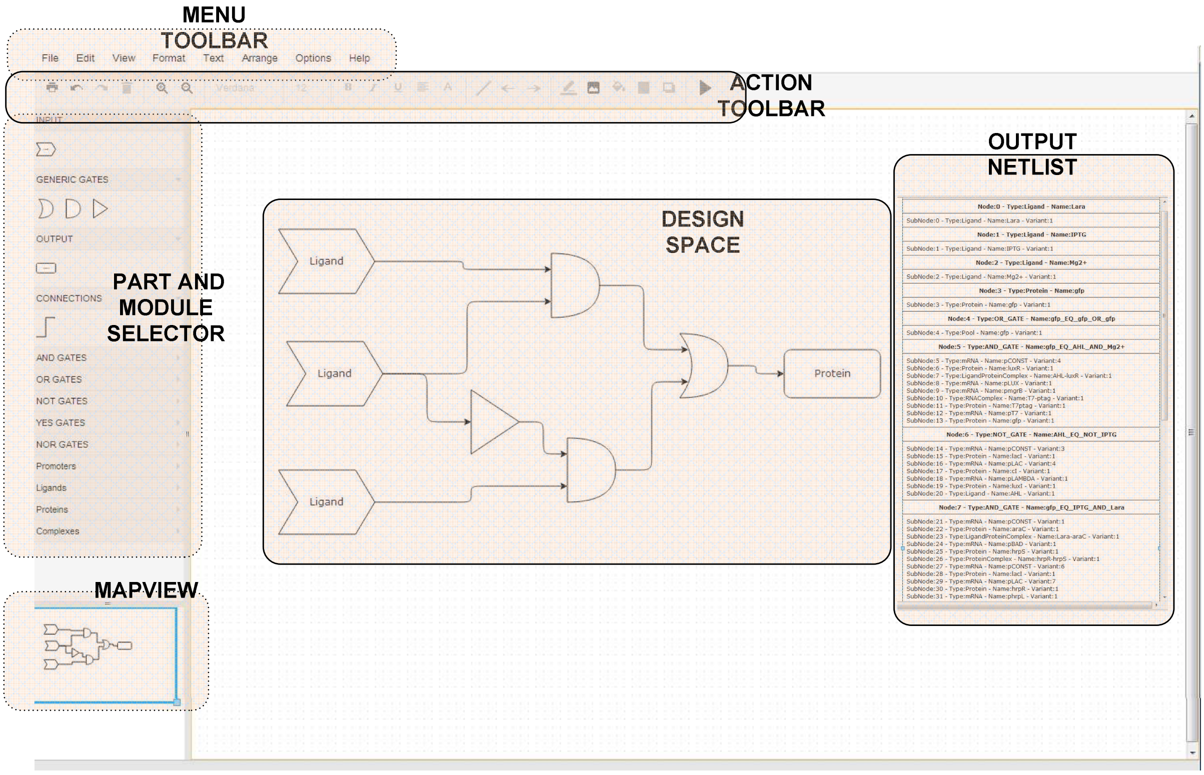


**Figure 1.** The workflow of SBROME within the automated design framework.

## 3. GRAPHICAL USER INTERFACE AND WEB SERVICE

The SBROME source code, together with the supporting module database can be downloaded from http://tagkopouloslab.ucdavis.edu/software.html. In addition, we have created an online service of a javascript-based CAD environment that can be launched by a web-browser (Firefox and Chrome are currently supported). Figure 2 depicts the basic components of the SBROME GUI.



```
Node:0 - Type:Ligand - Name:Lara
SubNode:0 - Type:Ligand - Name:Lara - Variant:1
Node:1 - Type:Ligand - Name:IPTG
SubNode:1 - Type:Ligand - Name:IPTG - Variant:1
Node:2 - Type:Ligand - Name:Mg2+
SubNode:2 - Type:Ligand - Name:Mg2+ - Variant:1
Node:3 - Type:Protein - Name:gfp
SubNode:3 - Type:Protein - Name:gfp - Variant:1
Node:4 - Type:OR_GATE - Name:gfp_EQ_gfp_OR_gfp
SubNode:4 - Type:Pool - Name:gfp - Variant:1
Node:5 - Type:AND_GATE - Name:gfp_EQ_AHL_AND_Mg2+
SubNode:5 - Type:mRNA - Name:pCONST - Variant:4
SubNode:6 - Type:Protein - Name:luxR - Variant:1
SubNode:7 - Type:LigandProteinComplex - Name:AHL-luxR - Variant:1
SubNode:8 - Type:mRNA - Name:pLUX - Variant:1
SubNode:9 - Type:mRNA - Name:pmgrB - Variant:1
SubNode:10 - Type:RNAComplex - Name:T7-ptag - Variant:1
SubNode:11 - Type:Protein - Name:T7ptag - Variant:1
SubNode:12 - Type:mRNA - Name:pT7 - Variant:1
SubNode:13 - Type:Protein - Name:gfp - Variant:1
Node:6 - Type:NOT_GATE - Name:AHL_EQ_NOT_IPTG
SubNode:14 - Type:mRNA - Name:pCONST - Variant:3
SubNode:15 - Type:Protein - Name:lacI - Variant:1
SubNode:16 - Type:mRNA - Name:pLAC - Variant:4
SubNode:17 - Type:Protein - Name:cI - Variant:1
SubNode:18 - Type:mRNA - Name:pLAMBDA - Variant:1
SubNode:19 - Type:Protein - Name:luxI - Variant:1
SubNode:20 - Type:Ligand - Name:AHL - Variant:1
Node:7 - Type:AND_GATE - Name:gfp_EQ_IPTG_AND_Lara
SubNode:21 - Type:mRNA - Name:pCONST - Variant:1
SubNode:22 - Type:Protein - Name:araC - Variant:1
SubNode:23 - Type:LigandProteinComplex - Name:Lara-araC - Variant:1
SubNode:24 - Type:mRNA - Name:pBAD - Variant:1
SubNode:25 - Type:Protein - Name:hrpS - Variant:1
SubNode:26 - Type:ProteinComplex - Name:hrpR-hrpS - Variant:1
SubNode:27 - Type:mRNA - Name:pCONST - Variant:6
SubNode:28 - Type:Protein - Name:lacI - Variant:1
SubNode:29 - Type:mRNA - Name:pLAC - Variant:7
SubNode:30 - Type:Protein - Name:hrpR - Variant:1
SubNode:31 - Type:mRNA - Name:phrpL - Variant:1
```

**Figure 2**. SBROME Graphical User Interface

### 3.1 Starting the application and importing files

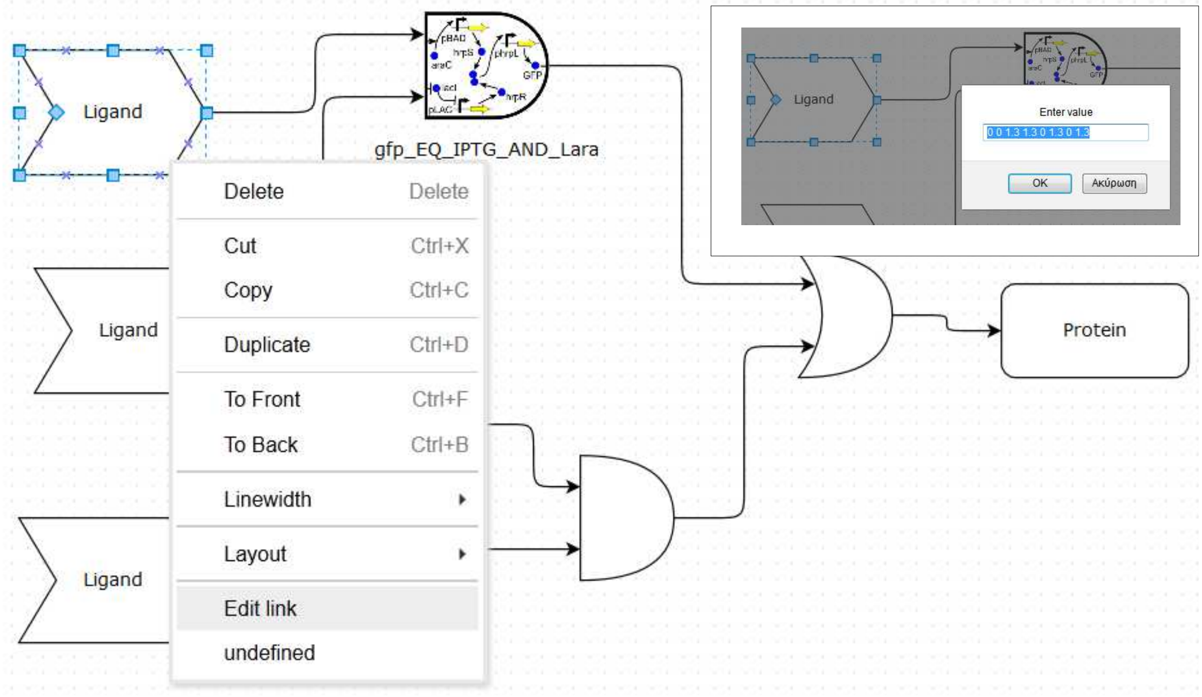
The web page must be started in a Firefox or Chrome browser if local files need to be imported. The GUI application can be directly launched from:

http://taglab1.cs.ucdavis.edu/htdocs/examples/grapheditor/www/index.html

Use **the File -> Import** command in the Menu and select one of the .XML files that are compatible with SBROME. In the software main page (http://tagkopouloslab.ucdavis.edu/software.html) we provide two sample XML files that serve as inputs for the multiplexer circuits in Figure 5 and 6 in [1]. More specifically, the file MULTIPLEXER_AND.XML is the AND-gate multiplexer design and the MULTIPLEXER_OR.XML file contains the OR-gate multiplexer design as they appear in [1].

SBROME GUI allows the user to use defined parts or modules for some nodes/elements in the design. As an example, the XML files MULTIPLEXER_AND_SPECIFIC.XML and MULTIPLEXER_OR_SPECIFIC.XML contain input files with the AND and OR gates have been replaced by specific topological designs, as reported in Table 1 of [1]. The design of both multiplexer arrangements can be replicated in the program by adding elements from the library bar in the left side of the application.

Currently, each design must contain at least one input and one output. The input and output nodes must be edited so that they contain the input/output states desired for the circuit. This is accomplished by selecting a node with left mouse click and pressing the “Edit Link” option, then inserting the state concentrations one by one with spaces (e.g. 0 1.2 0 1.2 0 0 0 0), as shown in Figure 3A and B.



Enter value

0 0 1.3 1.3 0 1.3 0 1.3

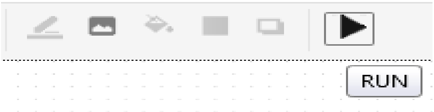
**Figure 3.** Entering the I/O characteristics of the circuit. Its input (here ligands) and output (here protein) nodes have to be populated with a vector of values.

The connection between nodes is done by pressing the connection icon and dragging a connection from the source node to the destination node, the positioning of the destination connection on the node is selected in the highlighted target node. The target node must be highlighted, in order to both allow a connection and select the connection place. Multiple connections from the same source must start at the exact same spot on the node as the first connection made.

The editor support Undo functions for any step (Ctrl+Z). The editor page must be refreshed (F5) before starting a new design, for the new design to work, especially after having received an output design.

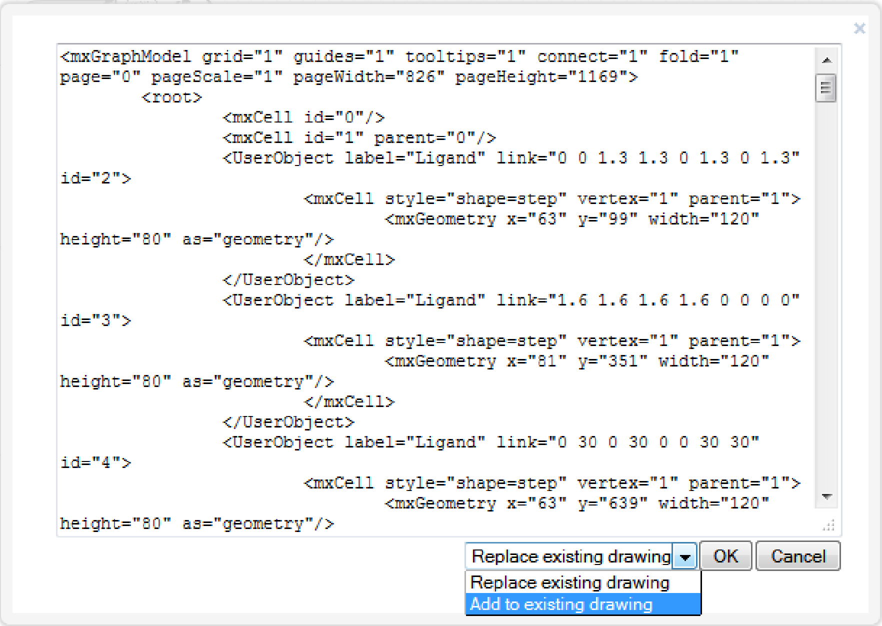
### 3.2 Running the optimization platform

In order to receive the output design for the specific loaded or designed input circuit, the RUN button must be pressed. It is located in the right most area of the toolbar, as shown below.



After pressing the RUN button, a window will appear that will provide the output design in

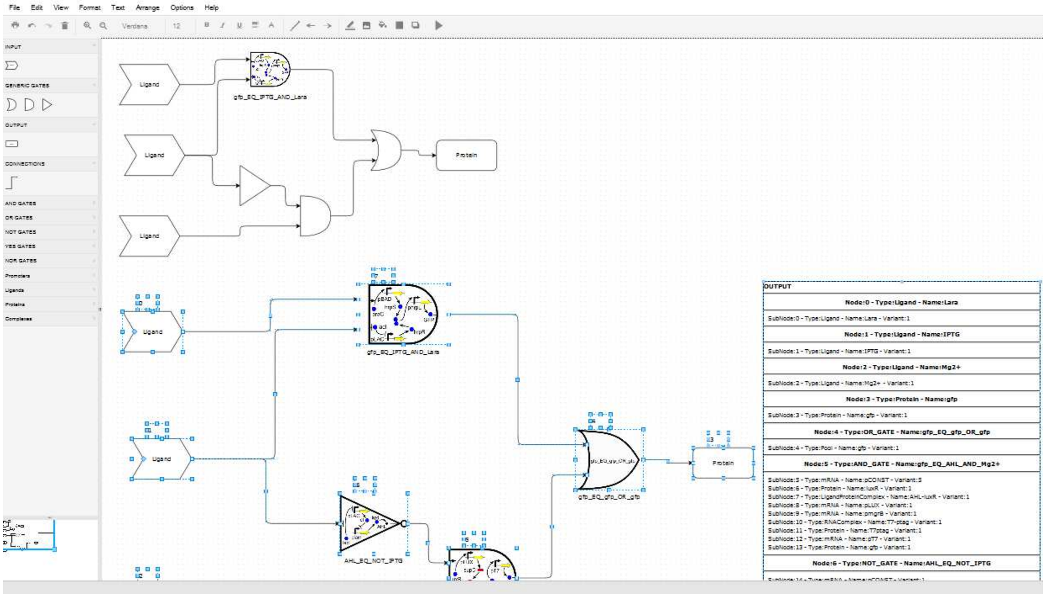
XML format and options to replace the input design or add the new design on top of it. The "Working…" message may appear, as it usually takes a few minutes to produce an output for the designs that are referenced in [1]. Figure 4 depicts the XML output after a successful run.



```
<mxGraphModel grid="1" guides="1" tooltips="1" connect="1" fold="1"
page="0" pageScale="1" pageWidth="826" pageHeight="1169">
        <root>
                <mxCell id="0"/>
                <mxCell id="1" parent="0"/>
                <UserObject label="Ligand" link="0 0 1.3 1.3 0 1.3 0 1.3"
id="2">
                        <mxCell style="shape=step" vertex="1" parent="1">
                                <mxGeometry x="63" y="99" width="120"
height="80" as="geometry"/>
                        </mxCell>
                </UserObject>
                <UserObject label="Ligand" link="1.6 1.6 1.6 1.6 0 0 0 0"
id="3">
                        <mxCell style="shape=step" vertex="1" parent="1">
                                <mxGeometry x="81" y="351" width="120"
height="80" as="geometry"/>
                        </mxCell>
                </UserObject>
                <UserObject label="Ligand" link="0 30 0 30 0 0 30 30"
id="4">
                        <mxCell style="shape=step" vertex="1" parent="1">
                                <mxGeometry x="63" y="639" width="120"
height="80" as="geometry"/>
```

**Figure 4.** XML output of the SBROME optimization platform.

There are two options for visualizing the optimized circuit, the user can either add or replace it to the initially designed specification. If the first option in selected (add), the user must drag the output design outside the margins of the input design to be clearly visible. The output design is selected by default when the OK button is pressed after the "Add" option is selected (Figure 5 and 6).



OUTPUT

Node:0 - Type:Ligand - Name:Lara

SubNode:0 - Type:Ligand - Name:Lara - Variant:1

Node:1 - Type:Ligand - Name:IPTG

SubNode:1 - Type:Ligand - Name:IPTG - Variant:1

Node:2 - Type:Ligand - Name:Mg2+

SubNode:2 - Type:Ligand - Name:Mg2+ - Variant:1

Node:3 - Type:Protein - Name:gfp

SubNode:3 - Type:Protein - Name:gfp - Variant:1

Node:4 - Type:OR_GATE - Name:gfp_EQ_gfp_OR_gfp

SubNode:4 - Type:Pool - Name:gfp - Variant:1

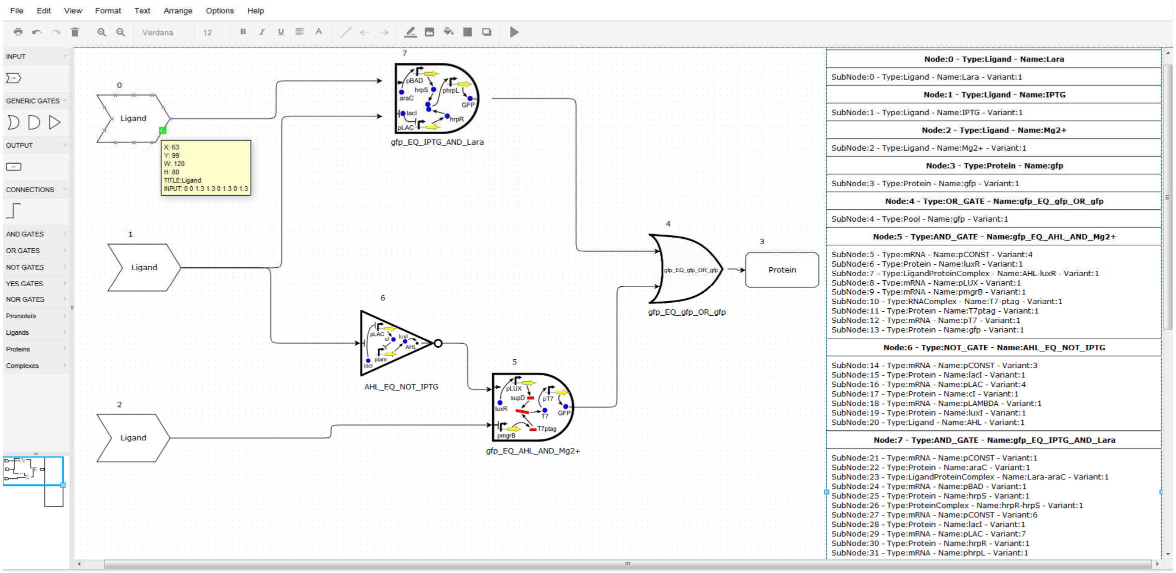
Node:5 - Type:AND_GATE - Name:gfp_EQ_AHL_AND_Mg2+

SubNode:5 - Type:mRNA - Name:pCONST - Variant:5
SubNode:6 - Type:Protein - Name:luxR - Variant:1
SubNode:7 - Type:LigandProteinComplex - Name:AHL-luxR - Variant:1
SubNode:8 - Type:mRNA - Name:pLUX - Variant:1
SubNode:9 - Type:mRNA - Name:pmgrB - Variant:1
SubNode:10 - Type:RNAComplex - Name:T7-ptag - Variant:1
SubNode:11 - Type:Protein - Name:T7ptag - Variant:1
SubNode:12 - Type:mRNA - Name:pT7 - Variant:1
SubNode:13 - Type:Protein - Name:gfp - Variant:1

Node:6 - Type:NOT_GATE - Name:AHL_EQ_NOT_IPTG

SubNode:14 - Type:mRNA - Name:pCONST - Variant:1

**Figure 5.** The design after the output is added and dragged below the input design.



X: 63
Y: 99
W: 120
H: 80
TITLE:Ligand
INPUT: 0 0 1.3 1.3 0 1.3 0 1.3

**Node:0 - Type:Ligand - Name:Lara**

SubNode:0 - Type:Ligand - Name:Lara - Variant:1

**Node:1 - Type:Ligand - Name:IPTG**

SubNode:1 - Type:Ligand - Name:IPTG - Variant:1

**Node:2 - Type:Ligand - Name:Mg2+**

SubNode:2 - Type:Ligand - Name:Mg2+ - Variant:1

**Node:3 - Type:Protein - Name:gfp**

SubNode:3 - Type:Protein - Name:gfp - Variant:1

**Node:4 - Type:OR_GATE - Name:gfp_EQ_gfp_OR_gfp**

SubNode:4 - Type:Pool - Name:gfp - Variant:1

**Node:5 - Type:AND_GATE - Name:gfp_EQ_AHL_AND_Mg2+**

SubNode:5 - Type:mRNA - Name:pCONST - Variant:4
SubNode:6 - Type:Protein - Name:luxR - Variant:1
SubNode:7 - Type:LigandProteinComplex - Name:AHL-luxR - Variant:1
SubNode:8 - Type:mRNA - Name:pLUX - Variant:1
SubNode:9 - Type:mRNA - Name:pmgrB - Variant:1
SubNode:10 - Type:RNAComplex - Name:T7-ptag - Variant:1
SubNode:11 - Type:Protein - Name:T7ptag - Variant:1
SubNode:12 - Type:mRNA - Name:pT7 - Variant:1
SubNode:13 - Type:Protein - Name:gfp - Variant:1

**Node:6 - Type:NOT_GATE - Name:AHL_EQ_NOT_IPTG**

SubNode:14 - Type:mRNA - Name:pCONST - Variant:3
SubNode:15 - Type:Protein - Name:lacI - Variant:1
SubNode:16 - Type:mRNA - Name:pLAC - Variant:4
SubNode:17 - Type:Protein - Name:cI - Variant:1
SubNode:18 - Type:mRNA - Name:pLAMBDA - Variant:1
SubNode:19 - Type:Protein - Name:luxI - Variant:1
SubNode:20 - Type:Ligand - Name:AHL - Variant:1

**Node:7 - Type:AND_GATE - Name:gfp_EQ_IPTG_AND_Lara**

SubNode:21 - Type:mRNA - Name:pCONST - Variant:1
SubNode:22 - Type:Protein - Name:araC - Variant:1
SubNode:23 - Type:LigandProteinComplex - Name:Lara-araC - Variant:1
SubNode:24 - Type:mRNA - Name:pBAD - Variant:1
SubNode:25 - Type:Protein - Name:hrpS - Variant:1
SubNode:26 - Type:ProteinComplex - Name:hrpR-hrpS - Variant:1
SubNode:27 - Type:mRNA - Name:pCONST - Variant:6
SubNode:28 - Type:Protein - Name:lacI - Variant:1
SubNode:29 - Type:mRNA - Name:pLAC - Variant:7
SubNode:30 - Type:Protein - Name:hrpR - Variant:1
SubNode:31 - Type:mRNA - Name:phrpL - Variant:1

**Figure 6.** The output screen after replacing the input design.

In the right output panel the inner components of each gate can be seen and the node numbers correspond to the numbers above each node in the diagram. Hovering over a node will reveal its type and other properties (like inner components and their variant).

## 4. REFERENCES

[1] L. Huynh, A. Tsoukalas, M. Köppe, I.Tagkopoulos, "SBROME: A scalable optimization and module matching framework for automated biosystem design", under revision, ACS Synthetic Biology, 2012