# A *Tech*nology mapping tool for **Gene**tic Circuits

## A quick start guide v2.0

November, 2019

Department of Electrical and Computer Engineering
Dhanani School of Science and Engineering
Habib University Karachi

# Contents

## About **Gene***Tech*

**Gene***Tech* (extracted from **Gene**tic *Tech*nology mapping) is a tool which allows a user to generate genetic logic circuits only by specifying the logical function desired to be achieved in a living cell. It performs Boolean optimization, followed by synthesis and technology mapping using a library of genetic logic gates. The genetic logic gates library used in this work has been developed and tested in the laboratory by MIT and Boston University [1]. **Gene***Tech* takes the Boolean expression of a genetic circuit as input, and then first optimize it. Afterwards, it synthesizes the optimized Boolean expression into NOR-NOT form in order to construct the circuit using the real NOR/NOT gates available in the genetic gates library [1]. In the end, **Gene***Tech* performs technology mapping to generate all the feasible circuits, with different genetic gates, to achieve the desired logical behavior.

After performing all the operations to generate the possible circuits, it processes all the circuits one by one to generate their standard SBOL files, Logical Representation and SBOL Visual Representation, and now you will able to save all the desired circuits and their representations as well, to be used in future or import in any other software.

**Gene***Tech* is a java and python based tool and is now available in executable application format. In this short document, you will learn how to use **Gene***Tech* (v2.0) to develop genetic circuits.

Stochastic simulation algorithm (SSA) has been implemented to perform stochastic simulations of SBML models. Furthermore, D-VASim is also capable of simulating the deterministic behavior of a bio model by solving ordinary differential equations.

Latest version of **Gene***Tech* can be downloaded from http://bda.compute.dtu.dk/downloads/genetech/. The sample set of genetic Boolean expressions have also been included in the download package.

## Conventions

The following conventions appear in this document:

| | |
|---|---|
| 💡 | This icon denotes a tip, which notifies you to advisory information. |
| ⚠ | This icon denotes an alert, which notifies you to important information. |
| **bold** | Bold text denotes items that you must select or click or enter the value in the software, such as providing input Boolean expression. |
| *italic* | Italic text denotes the name of a folder or a file path. |
| ***bold and italic*** | Bold and italic text denotes the name of a file. |

## 1. Running **Gene***Tech*

In this section, you will learn how to run **Gene***Tech* and construct desired genetic circuits.

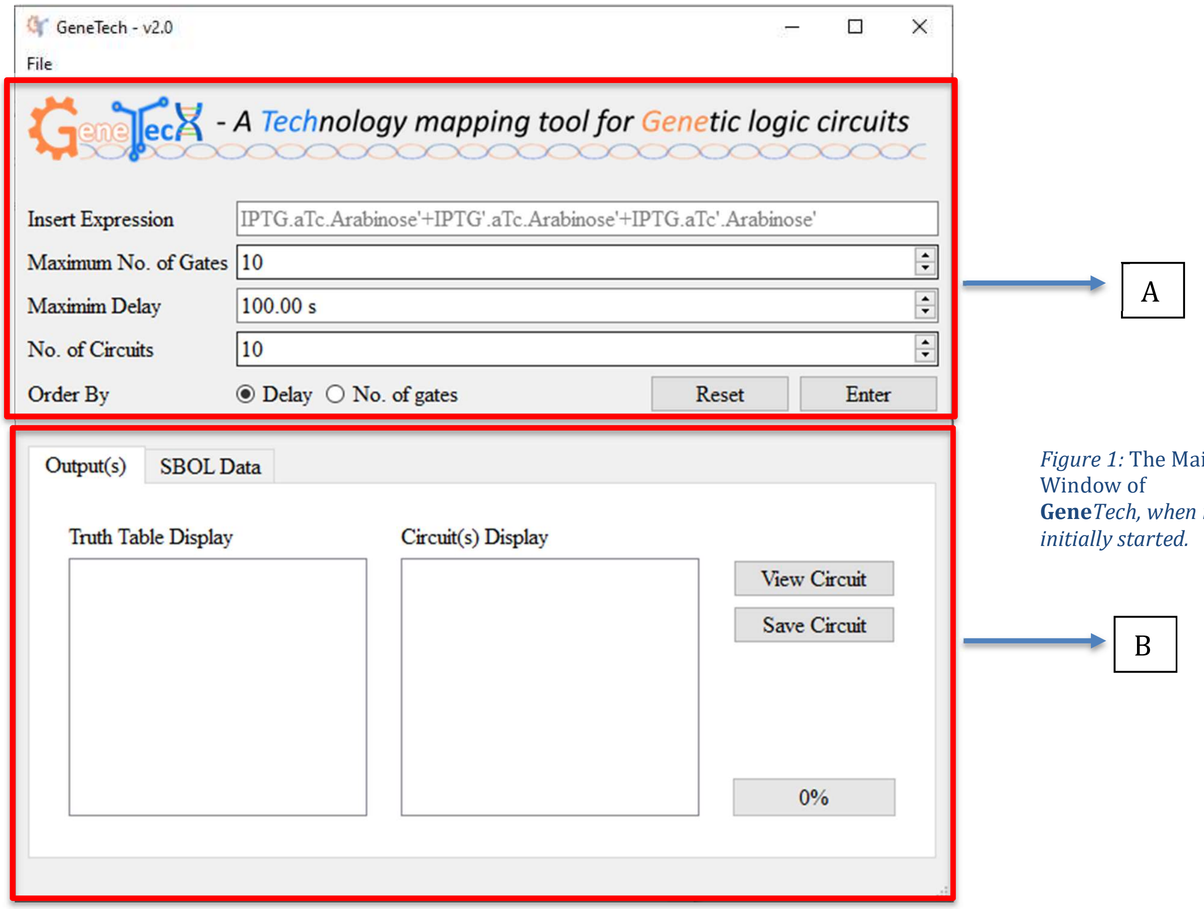
To run, **Gene***Tech*, first install the tool using the setup file, let all the required tools download and run Genetech.exe in the specified directory or shortcut Icon on Desktop.

**Alert** – Latest JAVA Runtime Engine (JRE) should be installed on your computer.

**Alert** – The screen-captured images included in this documentation are taken on Microsoft Windows.

### 1.1. Introduction to Graphic Interface of **Gene***Tech*

Once the tool is started, it appears as shown in Figure 1. The Main Window of the tool contains multiple portions. Two boxes, A and B, are shown in Figure 1. In Box A, Insert Expression is entry box where user can insert an input expression, based on that **Gene***Tech* performs the further operations. A sample equation is shown in the entry box in



Insert Expression: IPTG.aTc.Arabinose'+IPTG'.aTc.Arabinose'+IPTG.aTc'.Arabinose'
Maximum No. of Gates: 10
Maximim Delay: 100.00 s
No. of Circuits: 10
Order By: [✓ Delay] No. of gates

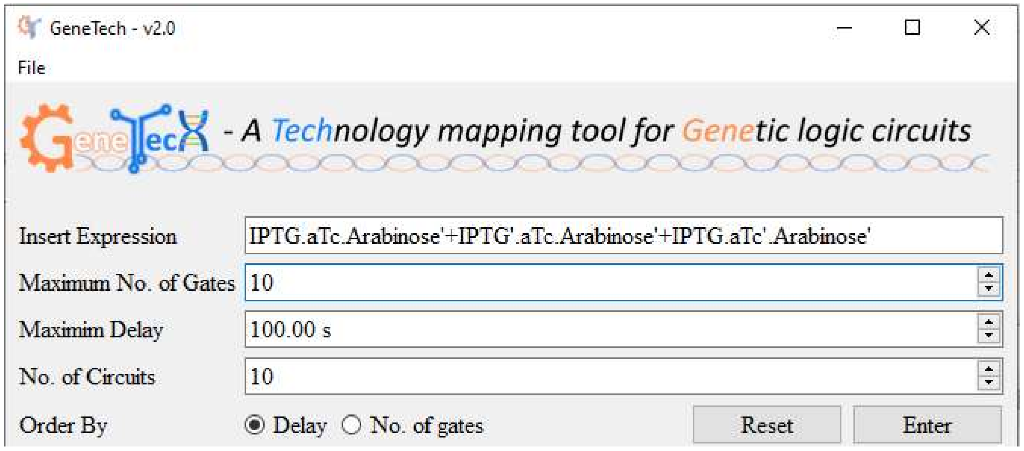
*Figure 1:* The Main Window of **Gene***Tech, when it is initially started.*

the form of a sum of products (SOP) expression which will vanish once user starts writing in the entry box. It also has the input entry for the maximum number of gates that is, how many maximum gates should be available in a generated circuit? For example, a Boolean expression inserted by the user is capable of generating 5 circuits. 2 of them are consist of 7 gates. So, if user sets the maximum number of gates as 6 or less then both of those circuits with 7 gates will not be shown. Similarly, a number inserted in the maximum delay box will filter all those circuits which have a delay equal to or less than the given number. A user can also sort the generated circuits on the basis of the delay of the circuits or based on the number of gates available in the circuits. In both cases the circuits are sorted in ascending order.

Box B shows two different lists views. The list view in the right represents the list where the both logic and SBOL visual representation of the circuits will be shown. The left list view will contain the truth table of the given Boolean expression. Example 1 in section 3.1 is a complete tutorial about using the tool.

## 1.2. Input Interface of Gene*Tech*

Once the tool is started, it first requires a user to provide inputs based on which the **Gene***Tech* perform optimization first. Since the tool first optimizes the Boolean expression based on the Simulated Annealing algorithm [2], it is therefore required to provide the values of **Maximum Number of Gates and Maximum Delay** (in seconds) along with the Boolean expression as shown in Figure 1.


GeneTech - v2.0
File
GeneTech - A Technology mapping tool for Genetic logic circuits
Reset
Enter

Insert Expression: IPTG.aTc.Arabinose'+IPTG'.aTc.Arabinose'+IPTG.aTc'.Arabinose'
Maximum No. of Gates: 10
Maximim Delay: 100.00 s
No. of Circuits: 10
Order By: [✓ Delay] No. of gates

*Figure 2: This figure shows that an input expression has been inserted in the window. Other inputs can be inserted accordingly.*

The Boolean function is inserted in the form of three variables **IPTG**, **aTc**, and **Arabinose**. These variables correspond to the name of external inducers and the logical operation is performed based on their presence/absence inside a living cell. **Gene***Tech* is able to process Boolean expression in both SOP and POS form. Figure 2 shows the screen-captured image of an instant when the user has entered a Boolean expression

**(IPTG'.aTc'.Arabinose'+IPTG'.aTc.Arabinose'+IPTG.aTc'.Arabinose')**.

It should be noted that there is no space in between the characters used in the Boolean expression.

**Alert –** The inverted terms should be described by " ' "; ANDed terms with " . "; and ORed terms with " + ", without any spaces.

**Alert –** Expression should be inserted in standard form. Standard SOP form is **a.b.c + a'b.c + a.b'.c** and the standard POS from is **(a+b+c).(a'b+c).(a+b'+c)**.

**Alert –** Names, spelling and the case of input inducers should remain the same i.e., IPTG cannot be iptg; aTc cannot be ATC or atc, etc.

## 1.2. Optimization and Synthesis by **Gene***Tech*

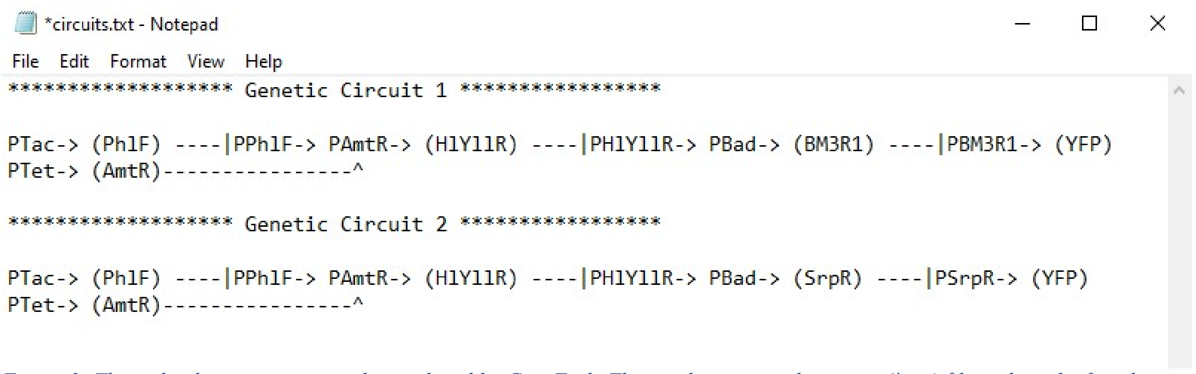
When a correct logic function is typed in, as shown in Figure 2, the tool first estimates the initial cost of a Boolean expression in terms of literals; where the name of each input inducer is considered as a separate literal. For example, the number of literals, in the entered Boolean expression are 9.

**Gene***Tech* then reduces the input Boolean expression and calculate its new cost, as shown in Figure 3. Once the input expression is reduced, it is then synthesized into NOR/NOT form, as depicted in the same figure.

## 1.2. Technology Mapping by **Gene***Tech*

Before performing technology mapping, the optimized expression is synthesized into NOR-NOT form because the genetic gates library used by **Gene***Tech* tool consists of NOR and NOT genetic gates. If **Gene***Tech* found compatible gates, it lists down all the possible genetic circuits in a file as shown in Figure 3. This figure shows that the tool suggests two possible different genetic circuits to obtain the logic expressed as a Boolean expression shown in Figure 2. **Gene***Tech*(v1.0) generated the circuit in the form of a text string which is similar to SBOL visual notations [3]. . **Gene***Tech*(v2.0) generates Logical and

SBOL visual notation as well. For example, the genetic circuit 1 shown in Figure 3 indicates that the protein, AmtR, generated (shown in the second line of circuit 1) by the promoter PTet represses its corresponding promoter PAmtR (shown in the first line of circuit 1). The same interaction can be observed in the SBOL notation shown in Figure 7. The Logical Representation of the same circuit shown in Figure 7 will help you understand this interaction in a better way.



```
******************* Genetic Circuit 1 *****************

PTac-> (PhlF) ----|PPhlF-> PAmtR-> (HlYllR) ----|PHlYllR-> PBad-> (BM3R1) ----|PBM3R1-> (YFP)
PTet-> (AmtR)----------------^

******************* Genetic Circuit 2 *****************

PTac-> (PhlF) ----|PPhlF-> PAmtR-> (HlYllR) ----|PHlYllR-> PBad-> (SrpR) ----|PSrpR-> (YFP)
PTet-> (AmtR)----------------^
```

*Figure 3: The technology mapping results produced by GeneTech. The results are saved in a text (*.txt) file and can be found in your path directory once the results are generated.*

## 1.3. File Generation by Gene*Tech*

After the file is saved with all the circuits represented in the of a text, the next steps are followed by reading those files by a python program. The text strings are read from a file and each circuit is separated in an array to process every component of the circuit accordingly. By gathering all the components, OR and NOR gates are formed depending on the number of inputs before a coding sequence. Output of one gate is the input for the next consecutive gate. In the case of NOR gate, second input is scrutinized to check from the gates library, shown in Figure 4, if the gate is Internal NOR gate or Semi-External NOR gate and the input is programmed accordingly. For example, NOR gate, (PPhlF, PAmtR, PHlYllR), in circuit 1 form Figure 3, is an Internal NOR gates and it requires its second input to be repressed from the output of the gate in line 2 of circuit 1, shown in SBOL notation in Figure 4.

A sample of SBOL file can be seen either by performing any example from this document or a small fraction can be seen in File Display section of Figure 10.

Logical Representation and SBOL visual notation are also worked in the same manner to generate images, which can be seen in Figure 7.

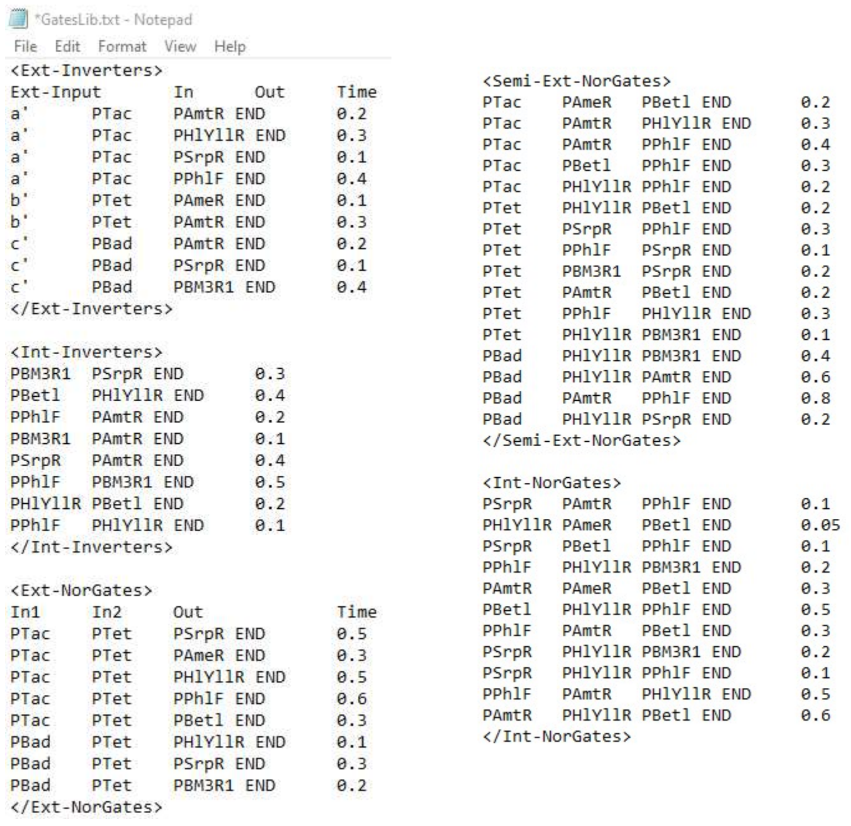
# 2. Genetic Gates Library

The genetic gates library used by **Gene***Tech* is available as a text file – ***GatesLib.txt*** and is shown in Figure 4. The gates are arranged in this file as Ext, Int or Semi-Ext-, based on if the inputs are external inputs i.e. IPTG (PTac) or aTc (PTet) or Arabinose (PBad); internal inputs; or semi-external inputs.

**Alert –** The ***GatesLib.txt*** file should not be removed from its default location inside the download folder, otherwise the tool will not function properly.

For example, in Figure 5, the first gate in the list of internal inverters, <Int-Inverters>, has an input promoter PBM3R1 which suppresses the output promoter PSrpR. Similarly, in the list of semi-external NOR gates, <Semi-Ext-NorGates>, the first input of the first NOR gate is one external input promoter i.e. PTac (or IPTG), the second one is intermediate promoter, PAmer, and they both suppress the output promoter PBetl.

Similarly, other repression-based gates can be added in the gates' library of **Gene***Tech*.



```
<Ext-Inverters>
Ext-Input       In      Out     Time
a'      PTac    PAmtR END       0.2
a'      PTac    PHlYllR END     0.3
a'      PTac    PSrpR END       0.1
a'      PTac    PPhlF END       0.4
b'      PTet    PAmeR END       0.1
b'      PTet    PAmtR END       0.3
c'      PBad    PAmtR END       0.2
c'      PBad    PSrpR END       0.1
c'      PBad    PBM3R1 END      0.4
</Ext-Inverters>

<Int-Inverters>
PBM3R1  PSrpR END       0.3
PBetl   PHlYllR END     0.4
PPhlF   PAmtR END       0.2
PBM3R1  PAmtR END       0.1
PSrpR   PAmtR END       0.4
PPhlF   PBM3R1 END      0.5
PHlYllR PBetl END       0.2
PPhlF   PHlYllR END     0.1
</Int-Inverters>

<Ext-NorGates>
In1     In2     Out             Time
PTac    PTet    PSrpR END       0.5
PTac    PTet    PAmeR END       0.3
PTac    PTet    PHlYllR END     0.5
PTac    PTet    PPhlF END       0.6
PTac    PTet    PBetl END       0.3
PBad    PTet    PHlYllR END     0.1
PBad    PTet    PSrpR END       0.3
PBad    PTet    PBM3R1 END      0.2
</Ext-NorGates>
```

```
<Semi-Ext-NorGates>
PTac    PAmeR   PBetl END       0.2
PTac    PAmtR   PHlYllR END     0.3
PTac    PAmtR   PPhlF END       0.4
PTac    PBetl   PPhlF END       0.3
PTac    PHlYllR PPhlF END       0.2
PTet    PHlYllR PBetl END       0.2
PTet    PSrpR   PPhlF END       0.3
PTet    PPhlF   PSrpR END       0.1
PTet    PBM3R1  PSrpR END       0.2
PTet    PAmtR   PBetl END       0.2
PTet    PPhlF   PHlYllR END     0.3
PTet    PHlYllR PBM3R1 END      0.1
PBad    PHlYllR PBM3R1 END      0.4
PBad    PHlYllR PAmtR END       0.6
PBad    PAmtR   PPhlF END       0.8
PBad    PHlYllR PSrpR END       0.2
</Semi-Ext-NorGates>

<Int-NorGates>
PSrpR   PAmtR   PPhlF END       0.1
PHlYllR PAmeR   PBetl END       0.05
PSrpR   PBetl   PPhlF END       0.1
PPhlF   PHlYllR PBM3R1 END      0.2
PAmtR   PAmeR   PBetl END       0.3
PBetl   PHlYllR PPhlF END       0.5
PPhlF   PAmtR   PBetl END       0.3
PSrpR   PHlYllR PBM3R1 END      0.2
PSrpR   PHlYllR PPhlF END       0.1
PPhlF   PAmtR   PHlYllR END     0.5
PAmtR   PHlYllR PBetl END       0.6
</Int-NorGates>
```

*Figure 4: The genetic gates library used by GeneTech. The gates are categorized bases on if the inputs to them are external, internal or semi-external.*

## 3. Examples

In this section, few very detailed and step by step examples are given for facilitating the user to run **Gene***Tech easily* and construct desired genetic circuits.

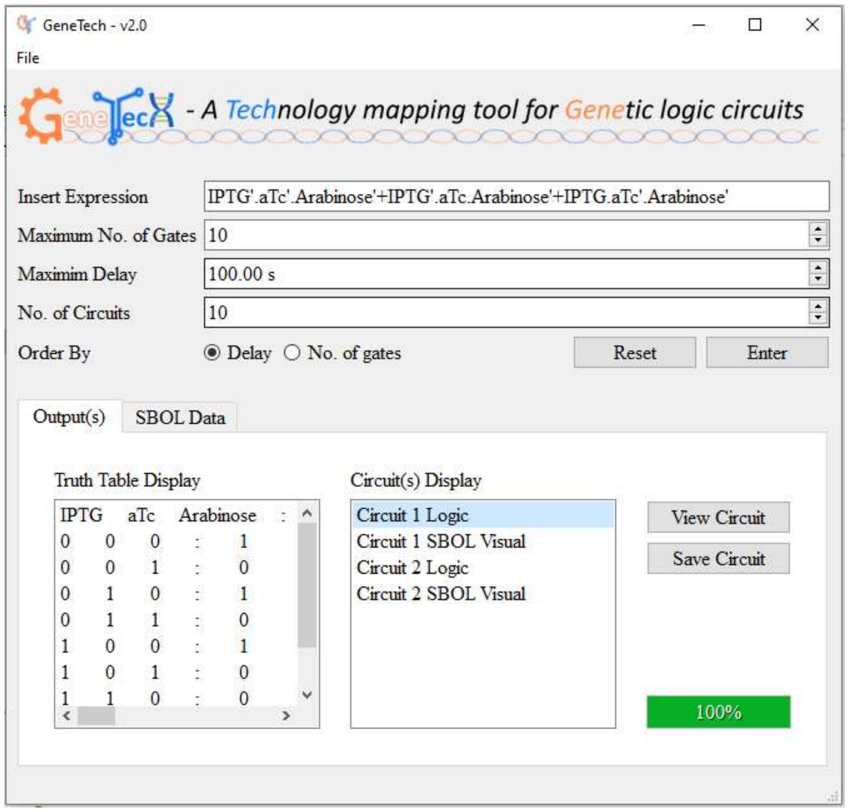
### 3.1. Example 1

Following steps have been followed to generate the genetic circuits for this specific example.

1. The input expression

   **IPTG'.aTc'.Arabinose'+IPTG'.aTc.Arabinose'+IPTG.aTc'.Arabinose'** is entered in the *Insert Expression* box given on the main window of the tool. Figure 2 shows that the same expression is inserted by the user
2. Insert the number of maximum gates and maximum delay accordingly. In this case the maximum number of gates are 10 and maximum delay is 10 ms. This means that all the circuits generated by this Boolean expression will have maximum 10 gates and the delay offered by each gate will be less than or equal to 10 ms.
3. Press **Enter**

Once the '**Enter'** is pressed, the tool will start generating the genetic circuits according to the process explained above. After the circuits are generated, the Main Window will look like Figure 5.


GeneTech - v2.0
File
GeneTech - A Technology mapping tool for Genetic logic circuits

Insert Expression: IPTG'.aTc'.Arabinose'+IPTG'.aTc.Arabinose'+IPTG.aTc'.Arabinose'
Maximum No. of Gates: 10
Maximim Delay: 100.00 s
No. of Circuits: 10
Order By: [✓ Delay] No. of gates

Truth Table Display

| IPTG | aTc | Arabinose | : | |
|---|---|---|---|---|
| 0 | 0 | 0 | : | 1 |
| 0 | 0 | 1 | : | 0 |
| 0 | 1 | 0 | : | 1 |
| 0 | 1 | 1 | : | 0 |
| 1 | 0 | 0 | : | 1 |
| 1 | 0 | 1 | : | 0 |
| 1 | 1 | 0 | : | 0 |

Circuit(s) Display
Circuit 1 Logic
Circuit 1 SBOL Visual
Circuit 2 Logic
Circuit 2 SBOL Visual

*Figure 5: This image shows the main window when a Boolean Expression has been entered and results are generated*

It can be seen that a truth table of the given Boolean expression has been generated under *Truth Table Display* and can be verified looking at given expression. There are two circuits generated by the given Boolean expression. Logic circuits and their respective SBOL visual forms have been shown under *SBOL* Visual *Logic Circuit(s) Display.* For example, Circuit 1: Logic is the logic circuit generated by the given expression and Circuit 1: SBOL Visual is the visual form of the same circuit. The .xml files generated for both the circuits have been shown under *SBOL Files Display*. The Logic Circuits and SBOL visual forms can be viewed by clicking on a specific circuit and then clicking the **View Circuit** Button given in the right bottom corner as shown in Figure 6. All the circuits generated by the input expression are shown in figure 7. The same logical components in the Logic form and SBOL form are generated with the same colour so that user can easily compare both forms of circuits.

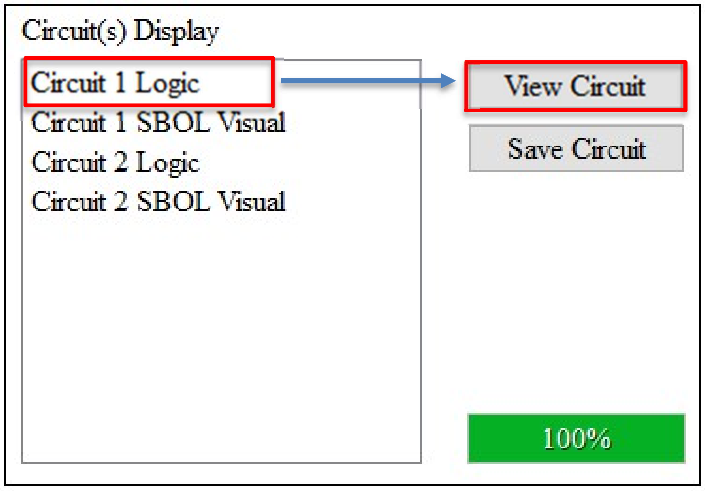
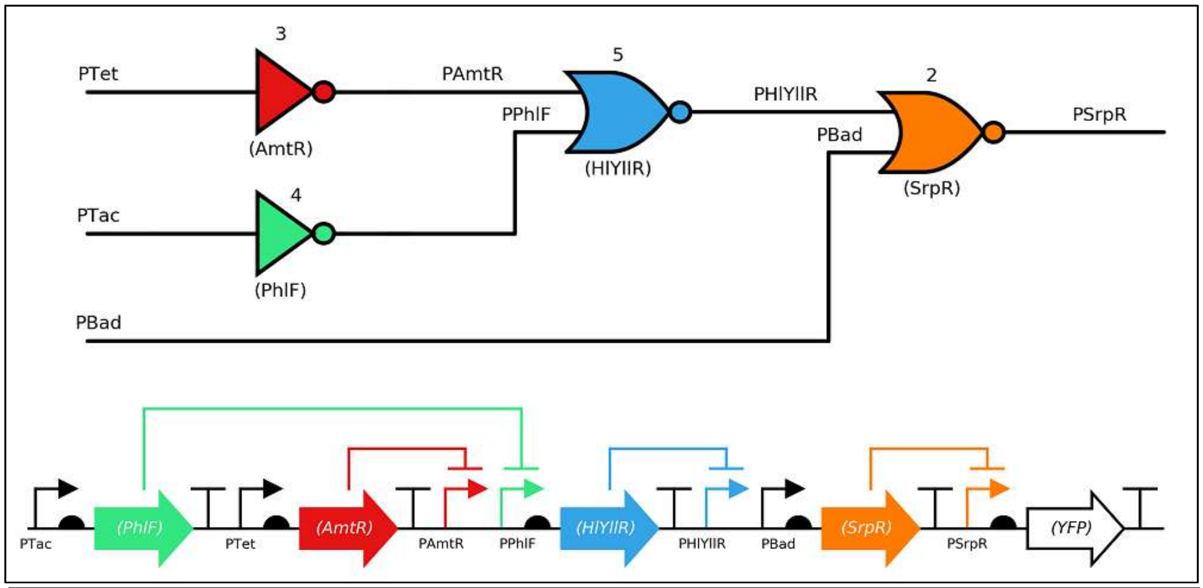


*Figure 6: This figure shows that how a circuit can be viewed by clicking the specific file name and press View Circuit Button.*

All of these circuits can be saved for the later use. To save a circuit, click on a circuit and then click the **Save Circuit** Button. Once the Save Circuit Button is clicked a Save Dialogue will appear where user can specify the path and the name of the file being saved. As an example, Circuit 1 Logic is shown to be saved in Figure 8.

**Alert** – A file name can't contain any of the following:
\ / : * ? < > | "@



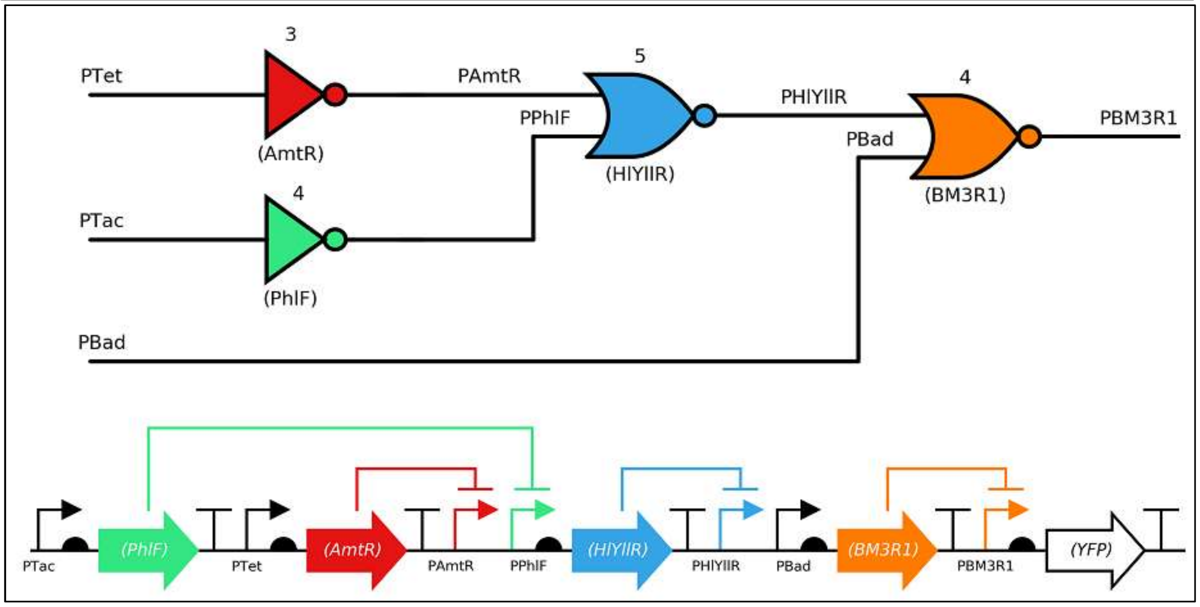


*Figure 7: This figure shows both the circuits generated by the given Boolean expression in both forms: Logic (upper) and SBOL Visual (bottom).*

For every circuit, a .xml file is also generated and shown under SBOL Data section. In this specific example SBOL File 1 is .xml file for Circuit 1 and SBOL File 2 is .xml file for Circuit 2, shown in figure 9. These files can be opened by clicking on a file. For example, on clicking **SBOL File 1**, the file opens under File Preview section as shown in Figure 9. Using Save As button, user can save the opened file as well. An already saved file can also be imported to view under **File Preview** via **Import File** button**.** Moreover, the same editing area can be used to take notes, if user prefers to do so, and to save the file.

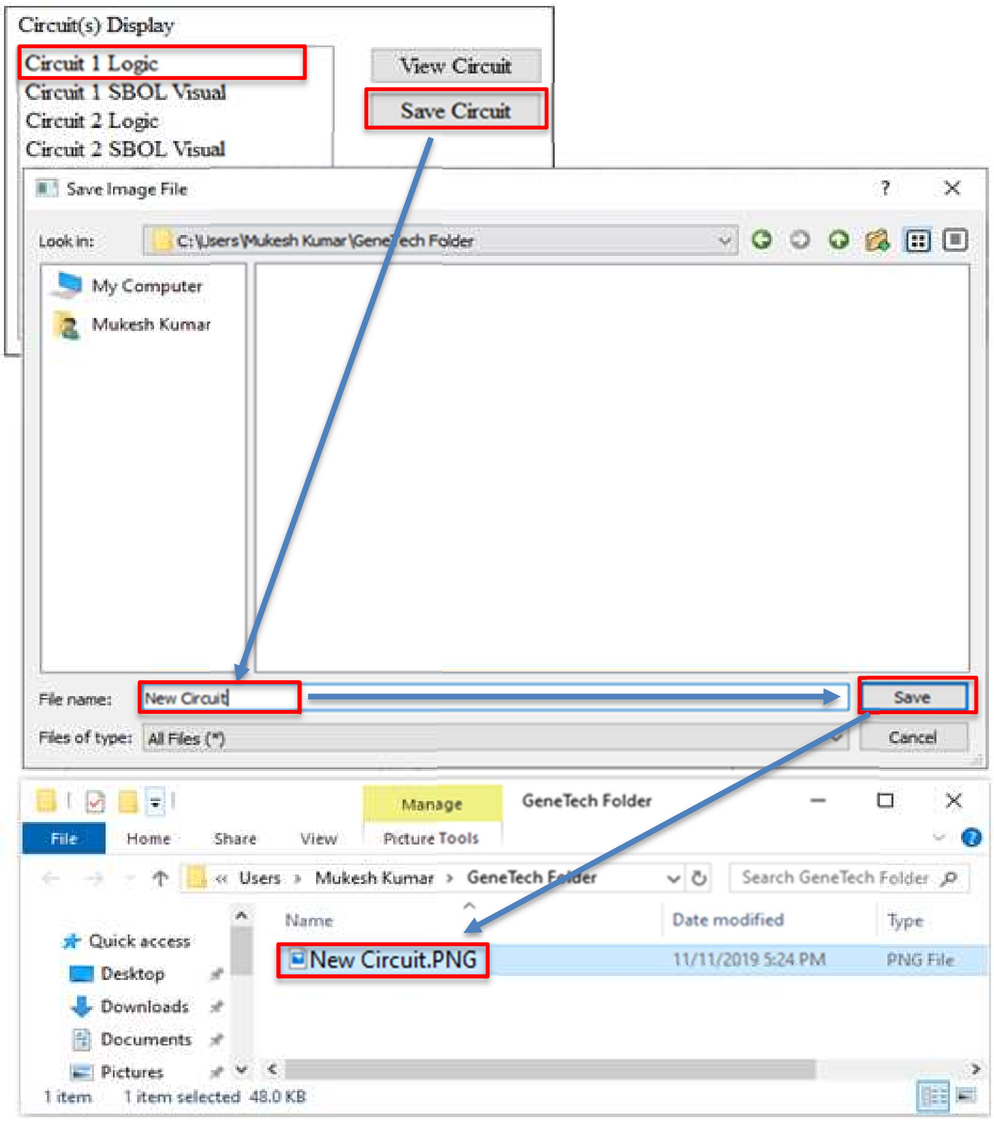
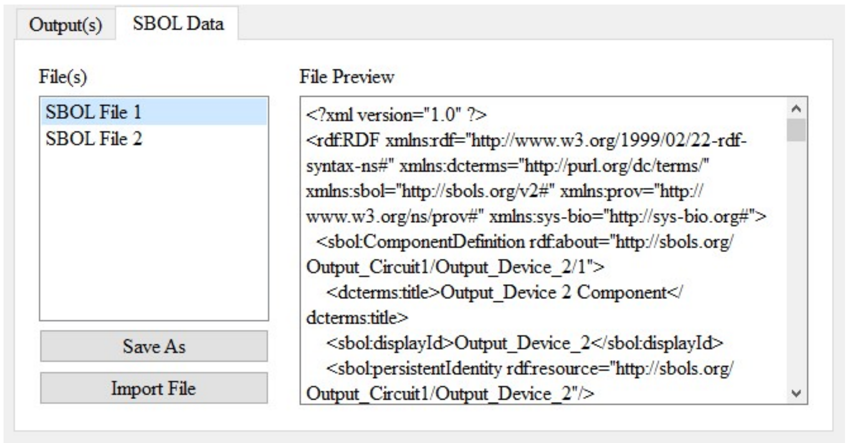


*Figure 8: This figure shows that a file is saved and shown in the directory.*



```
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#" xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:sbol="http://sbols.org/v2#" xmlns:prov="http://
www.w3.org/ns/prov#" xmlns:sys-bio="http://sys-bio.org#">
  <sbol:ComponentDefinition rdf:about="http://sbols.org/
Output_Circuit1/Output_Device_2/1">
    <dcterms:title>Output_Device 2 Component</
dcterms:title>
    <sbol:displayId>Output_Device_2</sbol:displayId>
    <sbol:persistentIdentity rdf:resource="http://sbols.org/
Output_Circuit1/Output_Device_2"/>
```

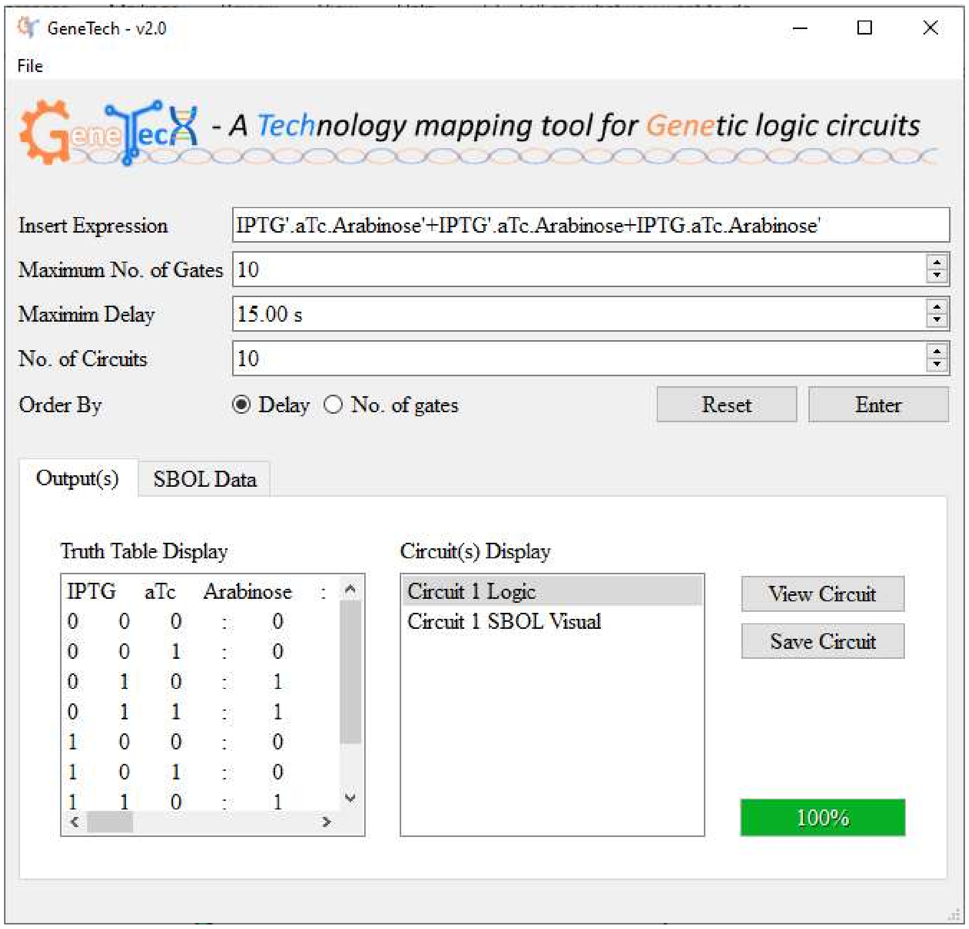
*Figure 9: In this figure generated SBOL data files are shown under File(s) and SBOL File 1 is opened under File Preview.*

## 3.2. Example 2

For this example the following steps are used.

1. Input Expression **IPTG'.aTc.Arabinose'+IPTG'.aTc.Arabinose+IPTG.aTc.Arabinose'** is entered in the Insert Expression box given on the main window of the tool.
2. Maximum number of gates in this example is set to 10 and maximum delay is set as 15 s, No. of gates are set as 10 and we want to order the circuits by the delay of the circuits.

Once the **Enter** button is pressed, the tool will start generating the genetic circuits according to the process explained above. After the circuits are generated, the Main Window will look like Figure 10 for this specific example. We can see that there is only one circuit generated by the given Boolean expression and other constraints. Generated circuit is shown in Figure 11.



Insert Expression: IPTG'.aTc.Arabinose'+IPTG'.aTc.Arabinose+IPTG.aTc.Arabinose'

Maximum No. of Gates: 10

Maximim Delay: 15.00 s

No. of Circuits: 10

Order By: [✓ Delay] No. of gates

Truth Table Display

| IPTG | aTc | Arabinose | : | |
|---|---|---|---|---|
| 0 | 0 | 0 | : | 0 |
| 0 | 0 | 1 | : | 0 |
| 0 | 1 | 0 | : | 1 |
| 0 | 1 | 1 | : | 1 |
| 1 | 0 | 0 | : | 0 |
| 1 | 0 | 1 | : | 0 |
| 1 | 1 | 0 | : | 1 |

Circuit(s) Display

Circuit 1 Logic
Circuit 1 SBOL Visual

*Figure 10: The image shows that with the given input expression, only 1 genetic circuit is created. It should be noted that that the maximum number of gates is 10 and maximum delay is 15 s, maximum allowed circuits could be generated 10 if possible. All the generated circuit has the delay less than or equal to 15s.*

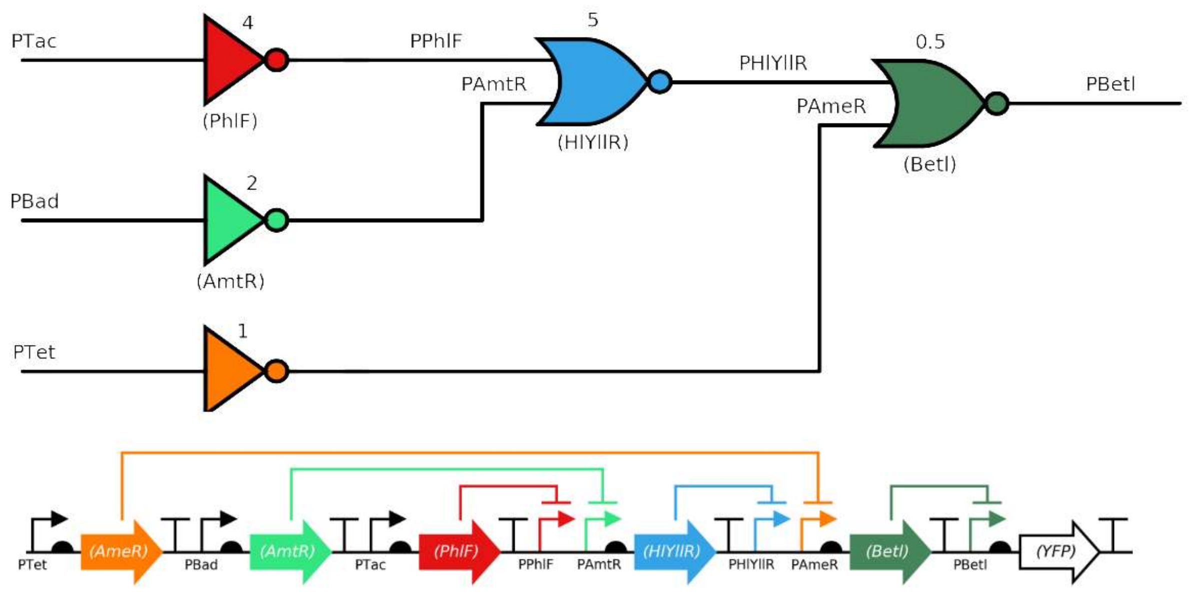


*Figure 11: At the top of this figure, Logic schematic is shown generated by the Boolean expression input in Figure 10. At the bottom the SBOL visual representation is given.*

# References

[1] A. a K. Nielsen, B. S. Der, J. Shin, P. Vaidyanathan, V. Paralanov, E. a Strychalski, D. Ross, D. Densmore, and C. a. Voigt, "Genetic circuit design automation.," *Science*, vol. 352, no. 6281, p. aac7341, 2016.

[2] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, "Optimization by Simulated Annealing," *Science (80-. ).*, vol. 220, no. 4598, pp. 671–680, 1983.

[3] J. Y. Quinn, R. S. Cox, A. Adler, J. Beal, S. Bhatia, Y. Cai, J. Chen, K. Clancy, M. Galdzicki, N. J. Hillson, N. Le Nov??re, A. J. Maheshwari, J. A. McLaughlin, C. J. Myers, P. Umesh, M. Pocock, C. Rodriguez, L. Soldatova, G. B. V Stan, N. Swainston, A. Wipat, and H. M. Sauro, "SBOL Visual: A Graphical Language for Genetic Designs," *PLoS Biol.*, vol. 13, no. 12, 2015.