Laboratory of Natural Information Processing
DA-IICT, Gandhinagar

Cello GUI
(Extending Cello with GUI)

USER MANUAL

Cello GUI
User Manual

Laboratory of Natural Information Processing
DA-IICT, Gandhinagar, Gujarat 382007
http://www.guptalab.org

Documentation version 1.0

Credits & Team

Principal Investigator: Dr. Manish K. Gupta
Graduate Mentor: Dixita Limbachiya
Developers: Vandit Modi, Smit Prajapati, Dhruv Patel, Vaibhav Amit Patel, Dhaval Parmar, Maharshi Vyas.

Acknowledgement

The authors would like to thank Cello Cidar Lab and Logisim for their open source softwares.

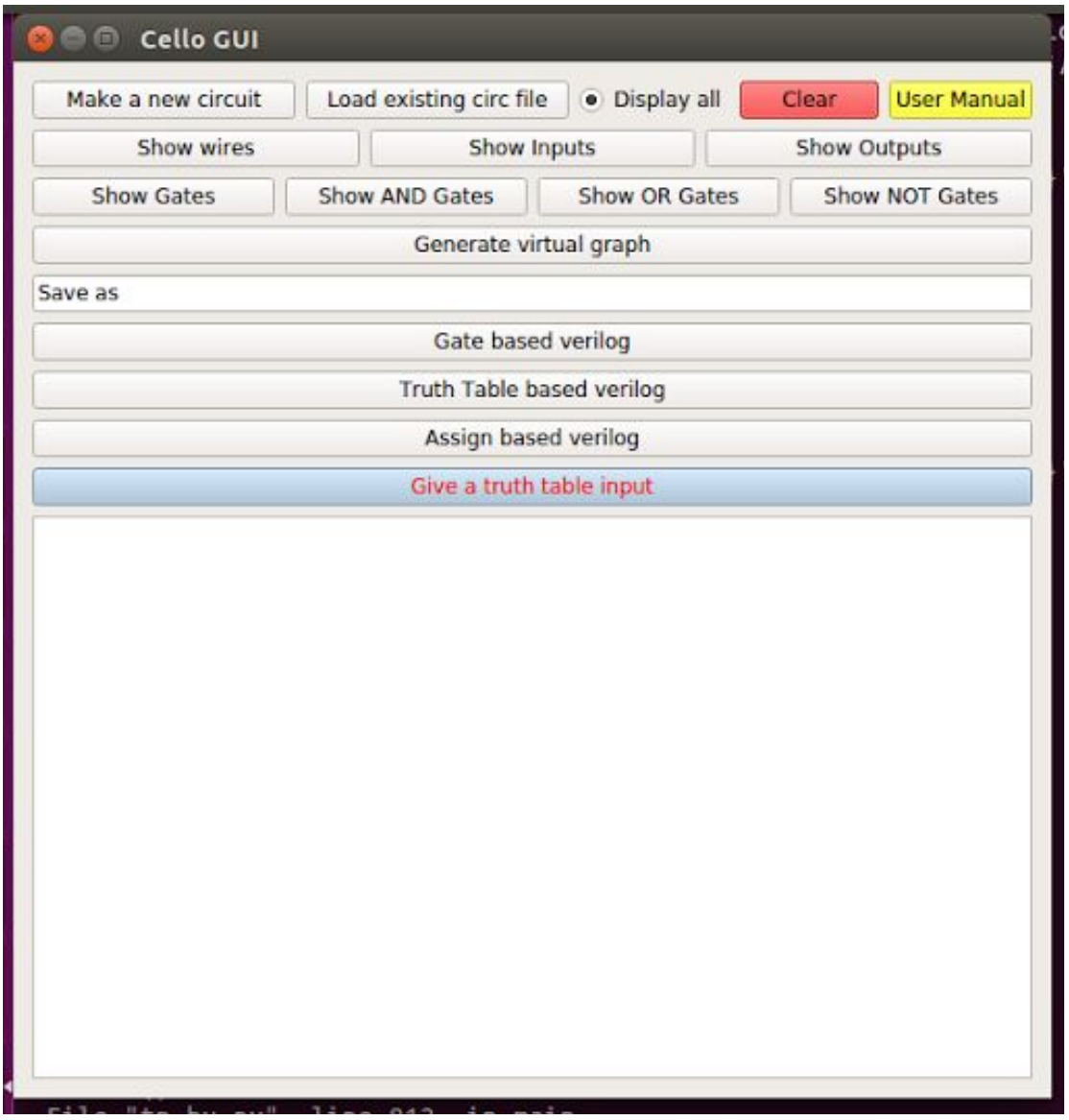
Table of contents

1. **Basics of Cello**
2. **Why to use Cello GUI?**
3. **Basic GUI**
4. **Main Module 1**
    a. Sub Module 1
    b. Sub Module 2
    c. Sub Module 3
5. **Main Module 2**
    a. Sub Module 1
6. **Legal Stuff**
7. **References/ Feedback**

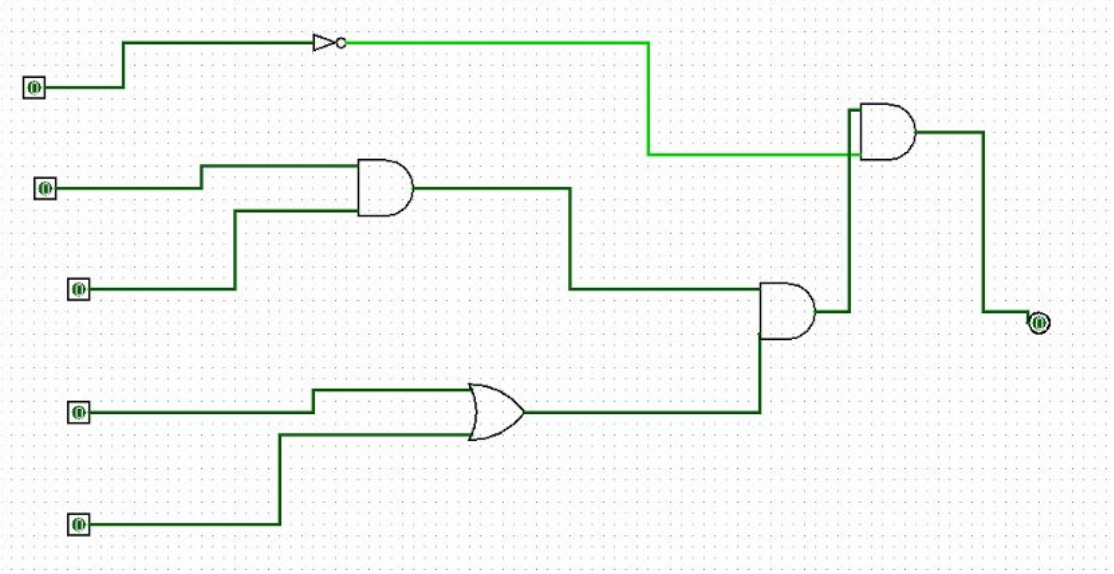
Basics of Cello

Why use Cello GUI?

Basic GUI parts



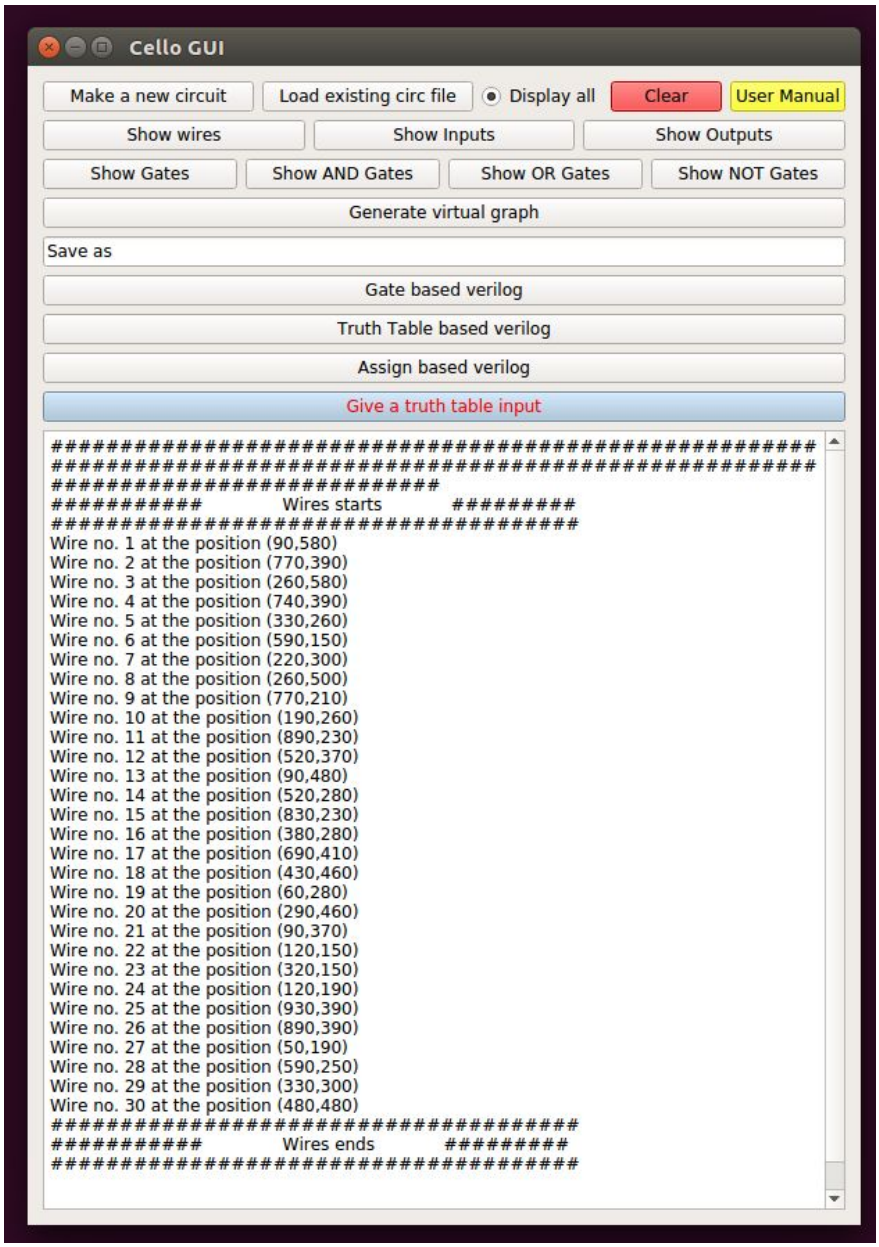
This is the main window of the software. The user can open logisim and draw circuits directly from the main window. After clicking on **Make a new circuit** user can open logisim or he/she can **Load an existing circ file.** There are two display mode, **display all** or disable display all. Because of the compact window of the software the sometimes the user has to scroll the output. That is why he or she can append the previous results or simply updating the screen every time. **Clear button** simply wipes the screen out.
User can read this **User Manual** directly using the yellow button from the GUI itself.



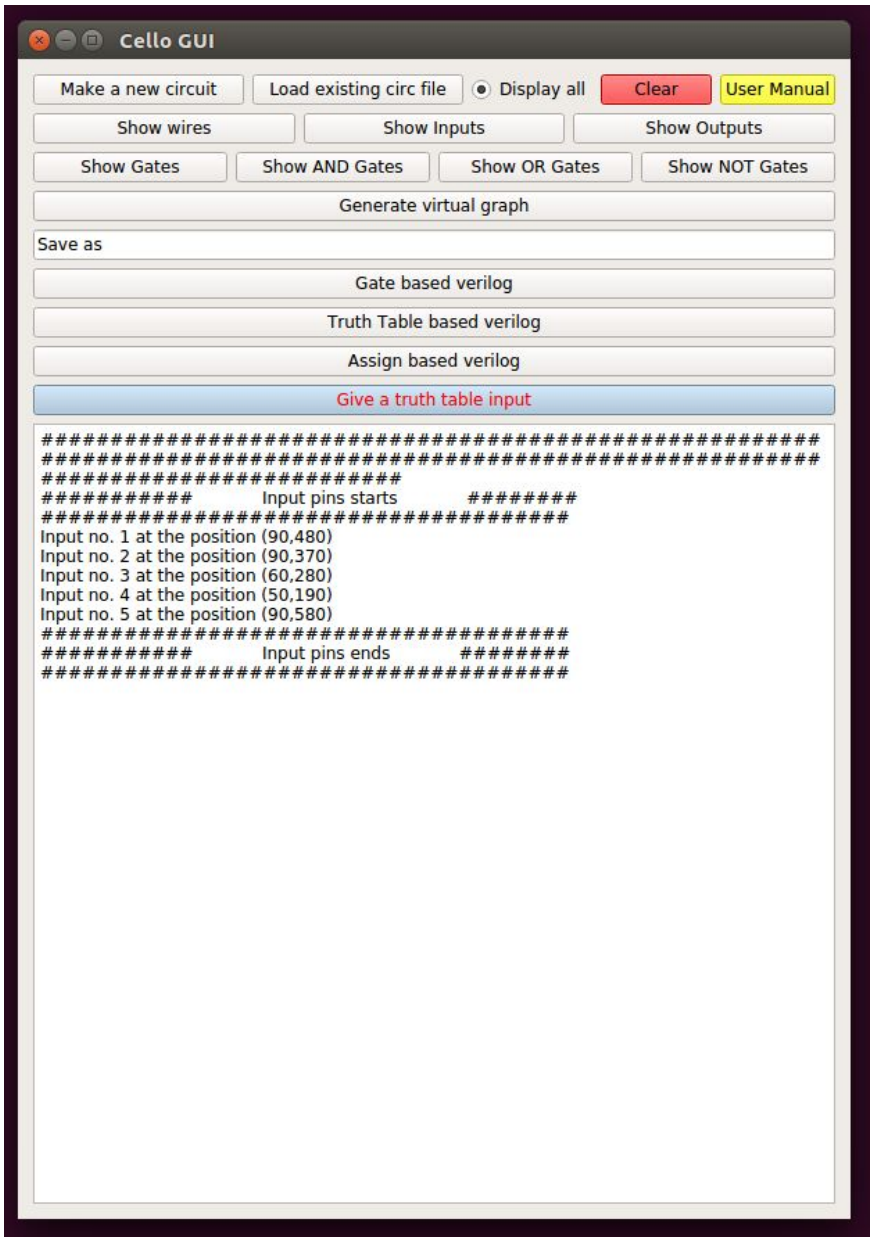
This circuit is created using logisim. Its circ file looks like the image below.

```
        <a name="font" val="SansSerif plain 12"/>
        <a name="halign" val="center"/>
        <a name="valign" val="base"/>
      </tool>
    </lib>
    <main name="main"/>
  - <options>
      <a name="gateUndefined" val="ignore"/>
      <a name="simlimit" val="1000"/>
      <a name="simrand" val="0"/>
    </options>
  - <mappings>
      <tool lib="6" map="Button2" name="Menu Tool"/>
      <tool lib="6" map="Button3" name="Menu Tool"/>
      <tool lib="6" map="Ctrl Button1" name="Menu Tool"/>
    </mappings>
  - <toolbar>
      <tool lib="6" name="Poke Tool"/>
      <tool lib="6" name="Edit Tool"/>
    - <tool lib="6" name="Text Tool">
        <a name="text" val=""/>
        <a name="font" val="SansSerif plain 12"/>
        <a name="halign" val="center"/>
        <a name="valign" val="base"/>
      </tool>
      <sep/>
    - <tool lib="0" name="Pin">
        <a name="tristate" val="false"/>
      </tool>
    - <tool lib="0" name="Pin">
        <a name="facing" val="west"/>
        <a name="output" val="true"/>
        <a name="labelloc" val="east"/>
      </tool>
      <tool lib="1" name="NOT Gate"/>
      <tool lib="1" name="AND Gate"/>
      <tool lib="1" name="OR Gate"/>
    </toolbar>
  - <circuit name="main">
      <a name="circuit" val="main"/>
      <a name="clabel" val=""/>
      <a name="clabelup" val="east"/>
      <a name="clabelfont" val="SansSerif plain 12"/>
      <wire from="(930,390)" to="(930,400)"/>
      <wire from="(220,300)" to="(220,370)"/>
      <wire from="(830,230)" to="(890,230)"/>
      <wire from="(590,250)" to="(780,250)"/>
      <wire from="(690,410)" to="(690,480)"/>
      <wire from="(290,460)" to="(290,480)"/>
      <wire from="(220,300)" to="(330,300)"/>
      <wire from="(260,500)" to="(430,500)"/>
```

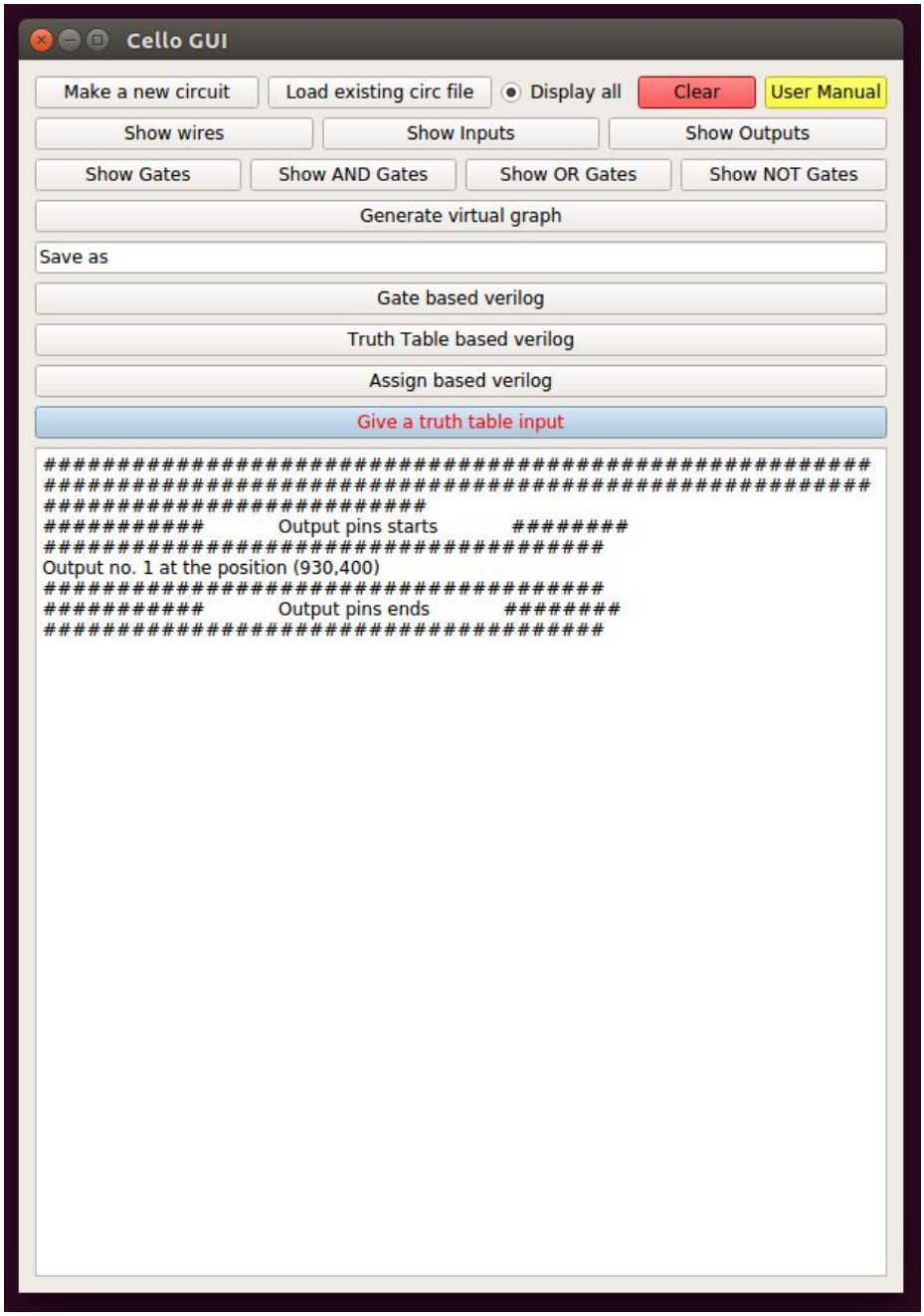
Show wires: Shows the extracted wires from the canvas.



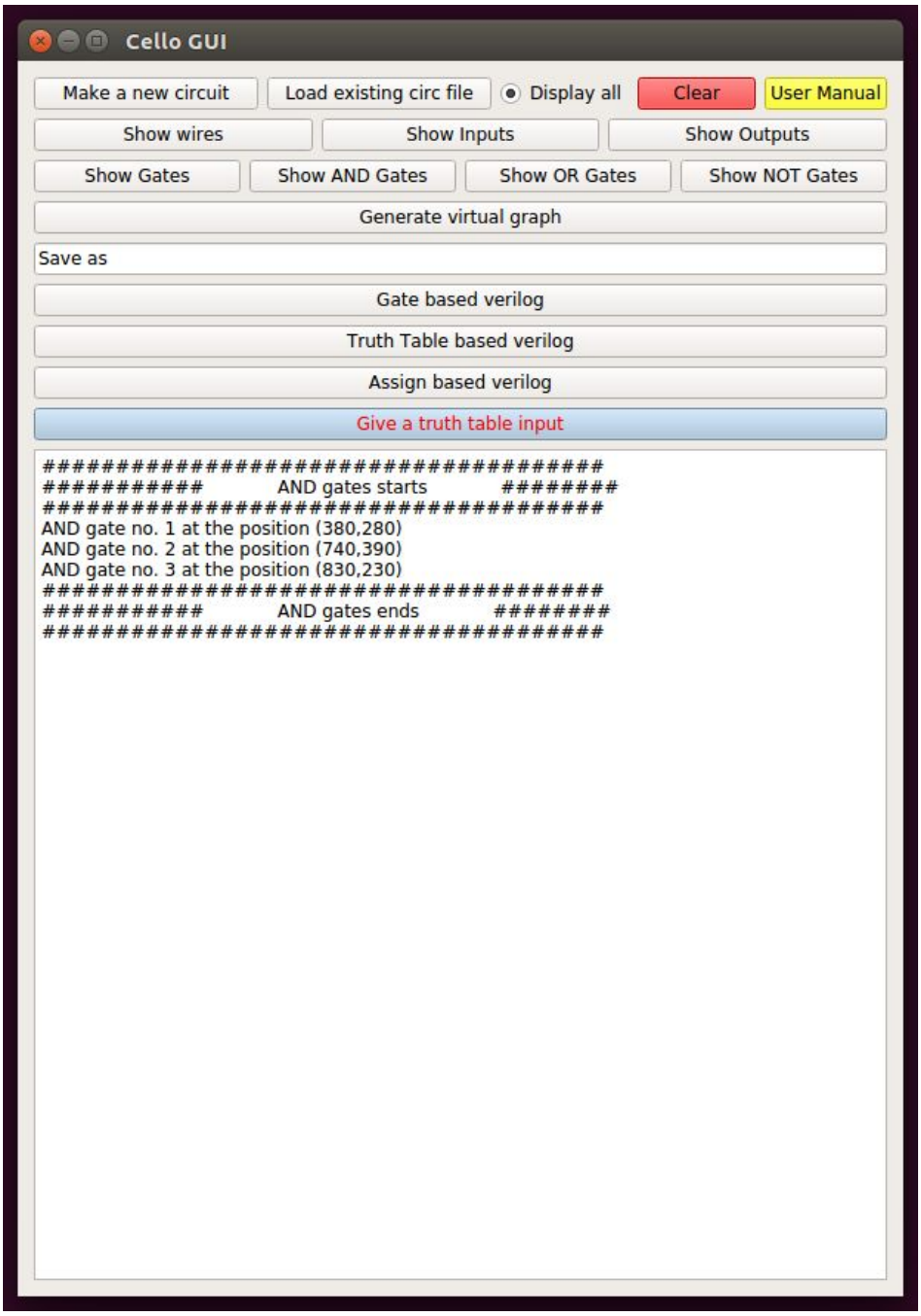
Show inputs: Shows the extracted inputs from the canvas.



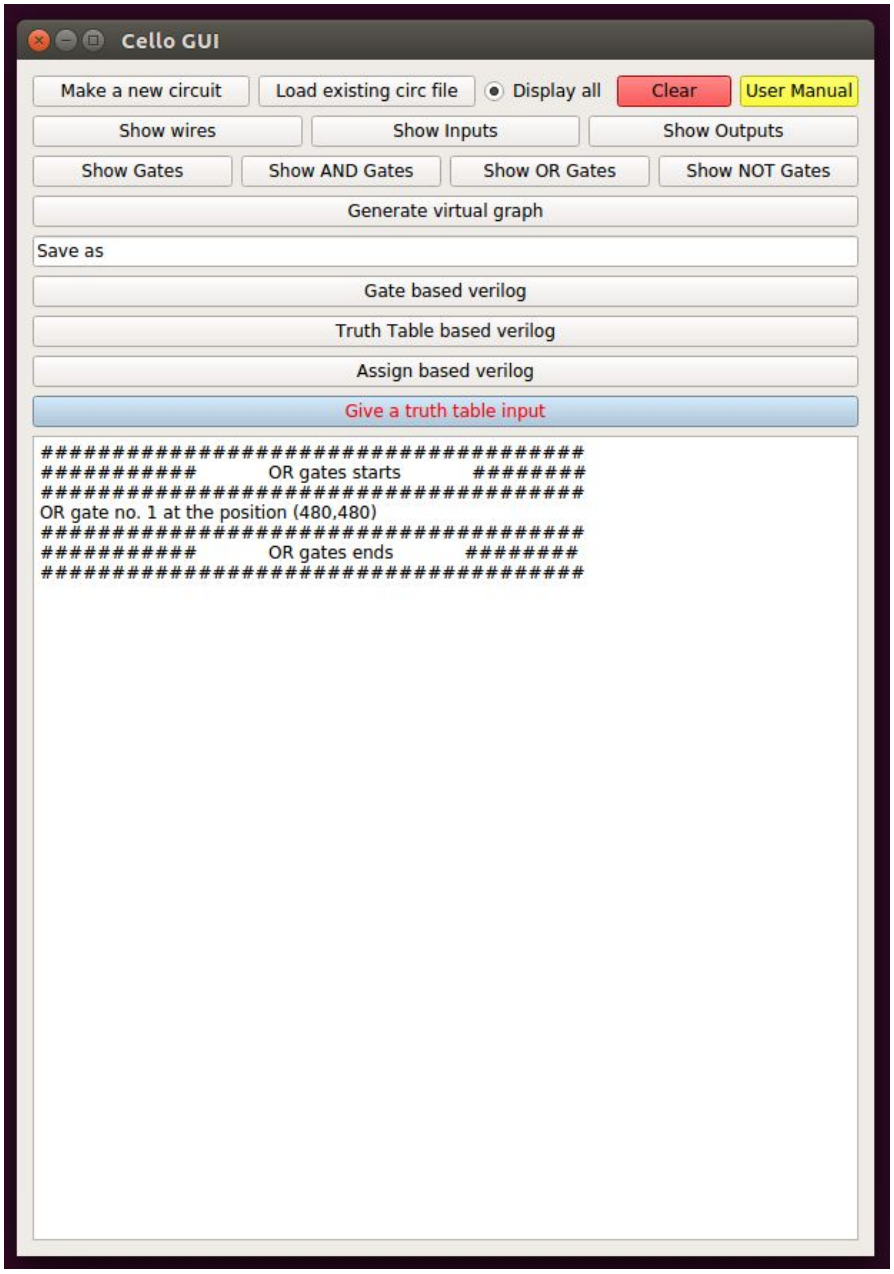
Show outputs: Shows the extracted output pins from the canvas.



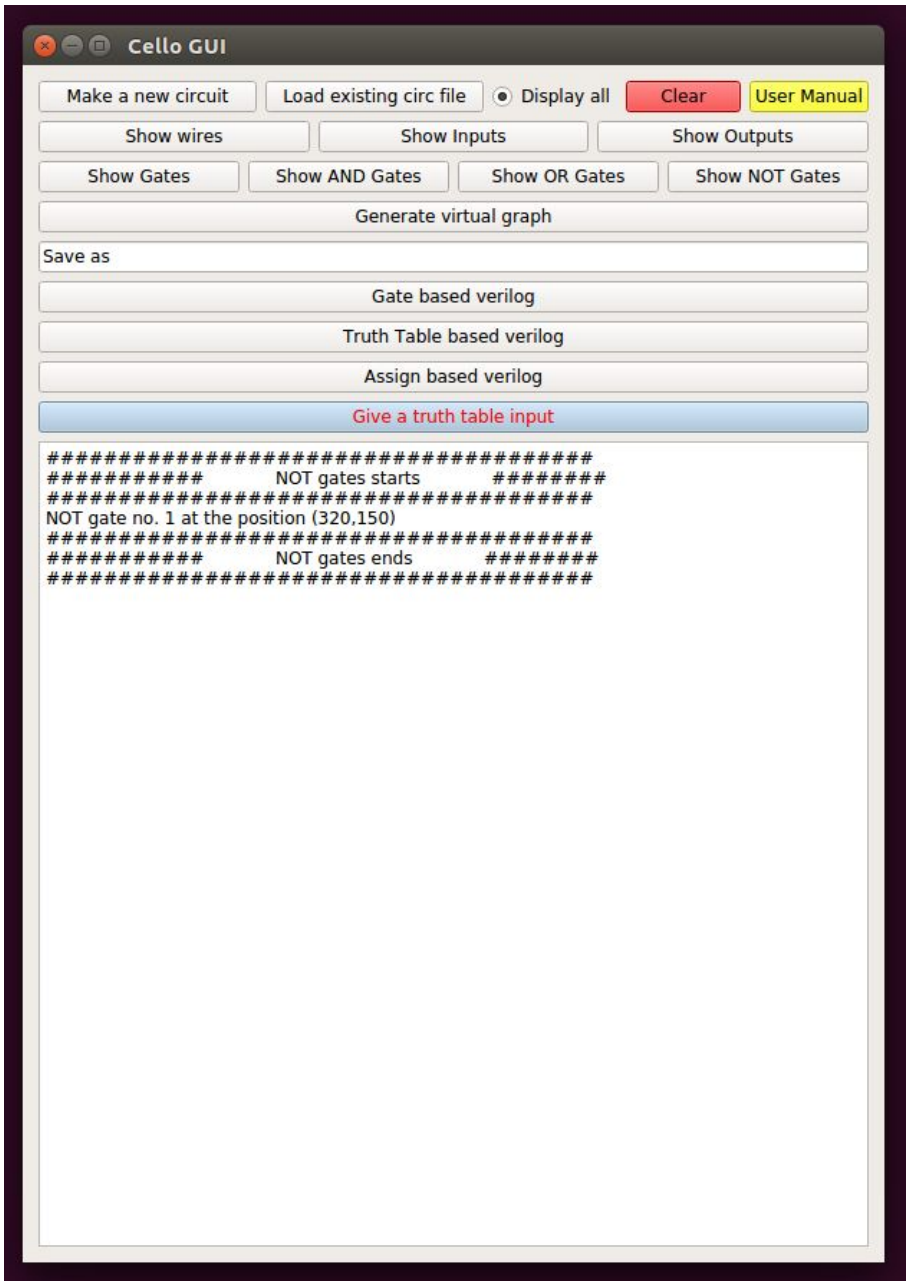
Show AND gates: Shows the position of AND gates extracted from the canvas.



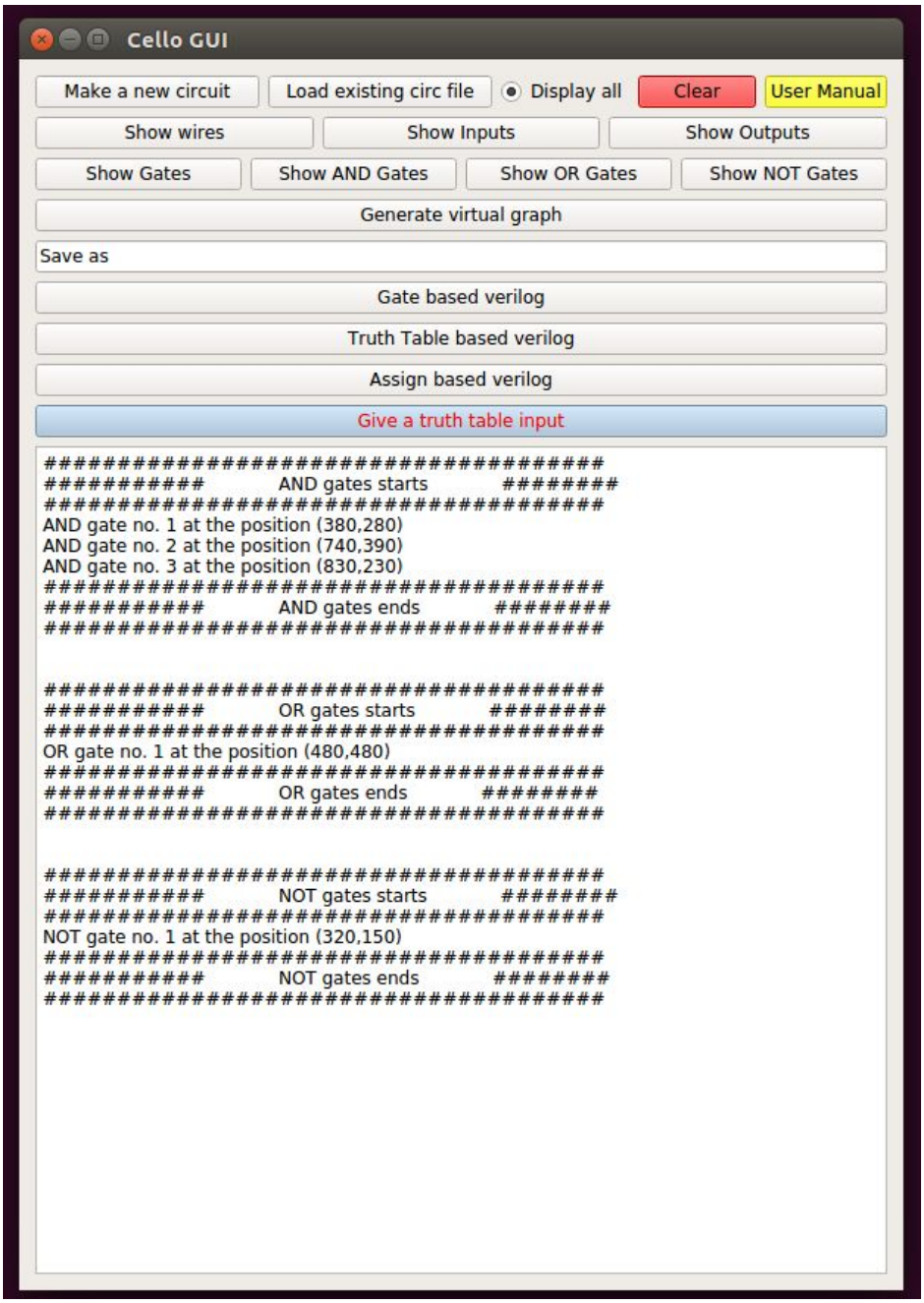
Show OR gates: Shows the position of OR gates extracted from the canvas.



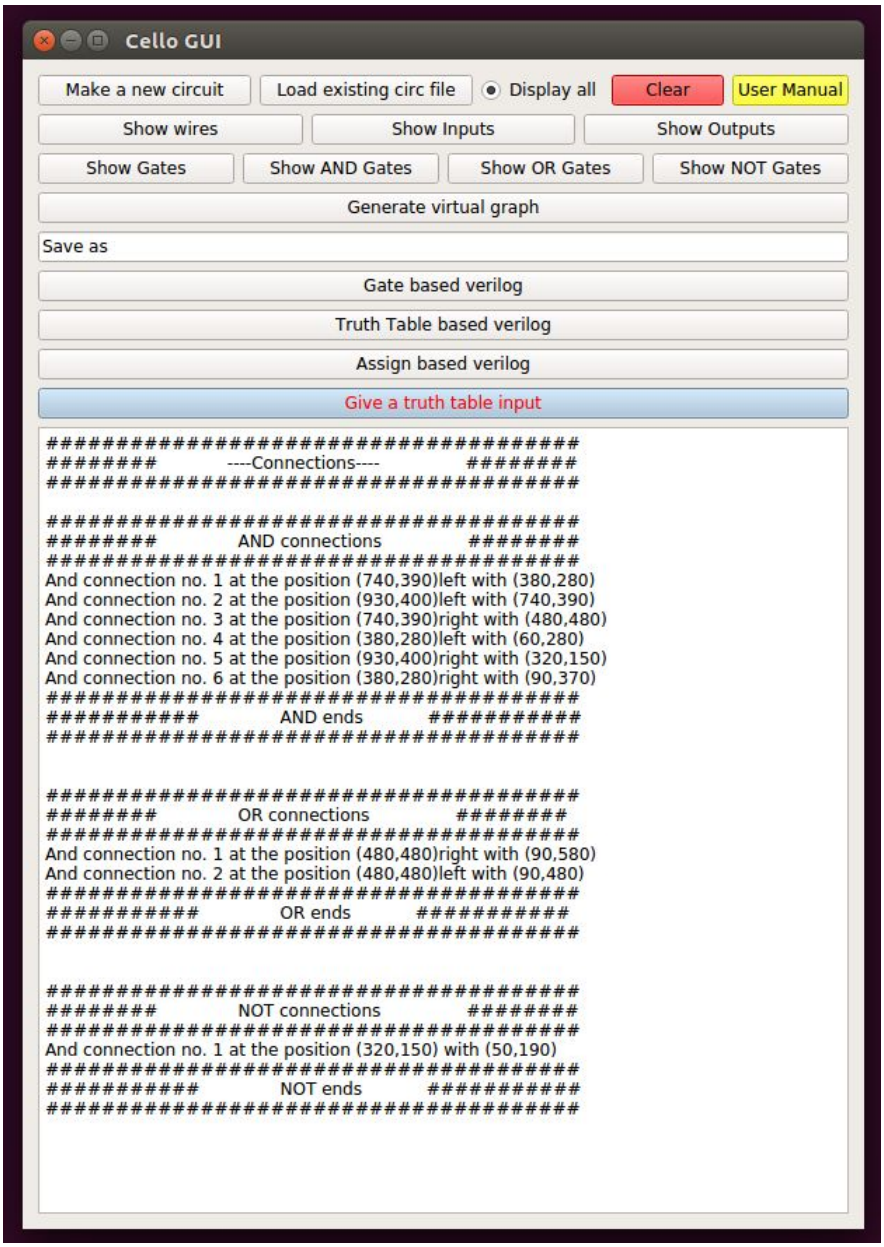
Show NOT gates: Shows the position of NOT gates extracted from the canvas.



Show Gates: Shows the position of All the gates extracted from the canvas.



Generate virtual graph: The software processes the information extracted and shown above and generates a virtual graph containing information about the connection between Inputs, Outputs, and Gates treating them as vertices and wires as edges.



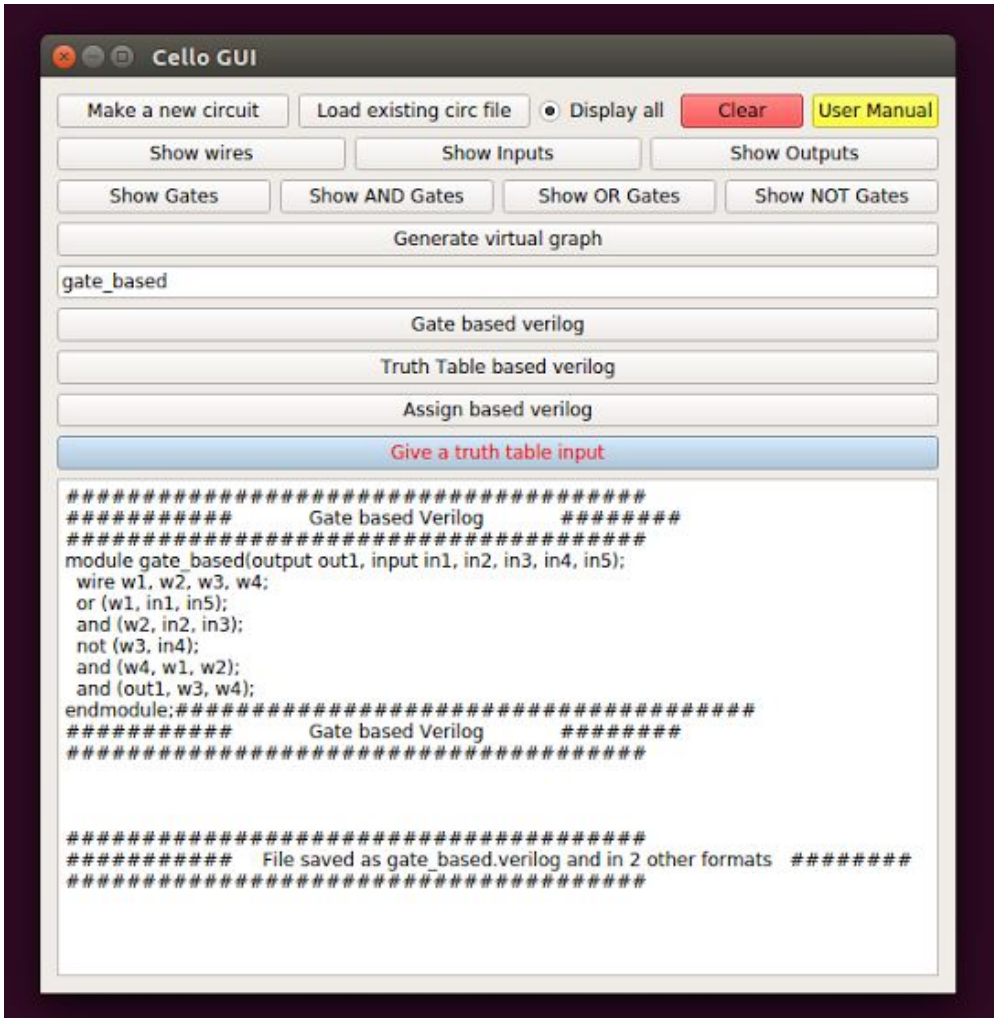
Main Module 1

The software provides first easy conversion from **Logisim** circ file to verilog file for **Cello**.

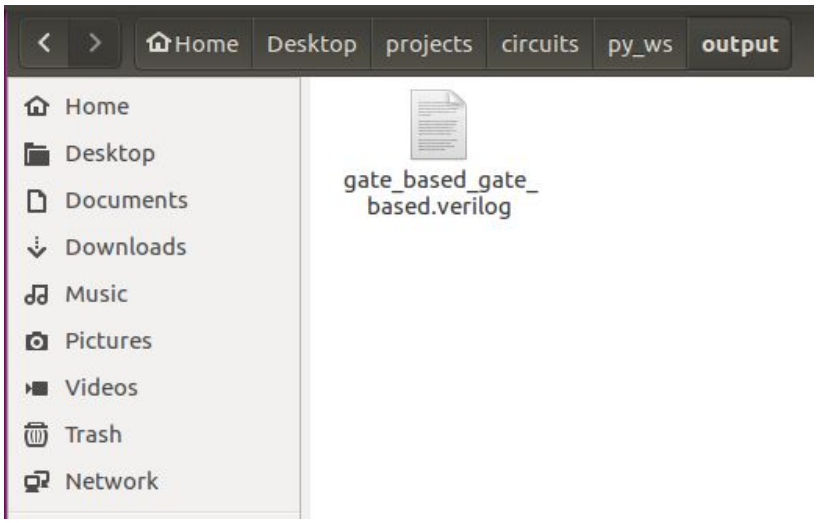
The First module contains three sub modules:

- Gate based verilog conversion
- Truth table based verilog conversion
- Assign based Verilog conversion

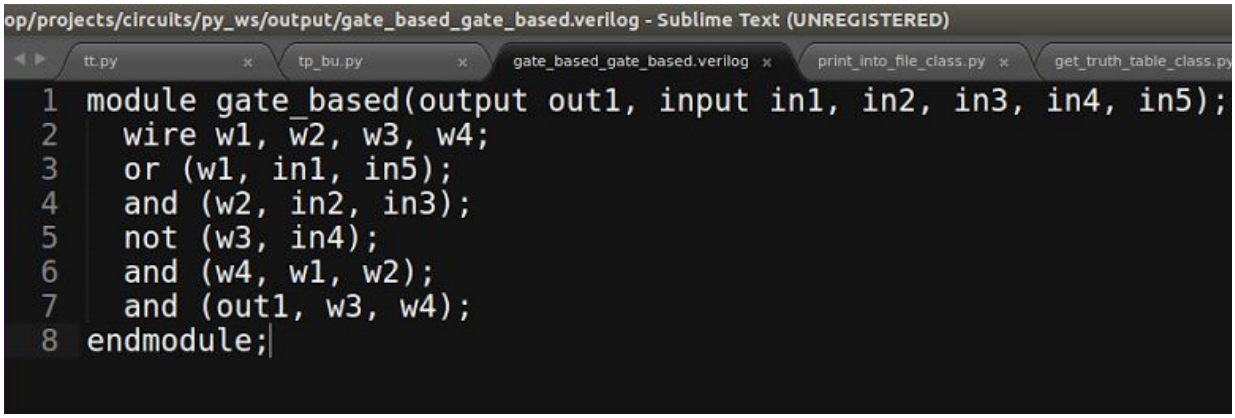
Gate based verilog conversion:



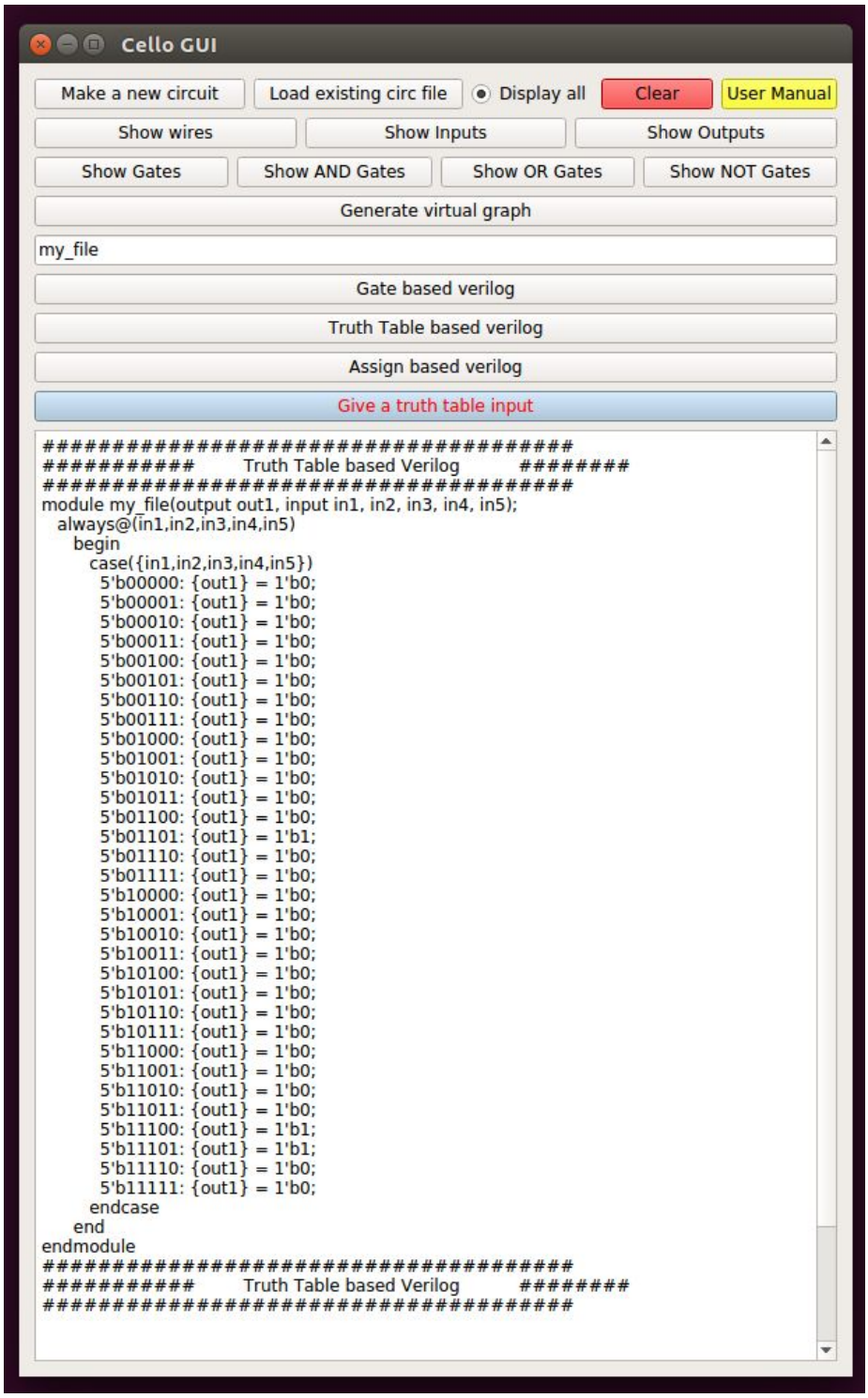
The software generates a verilog file named gate_based.verilog in the output directory.



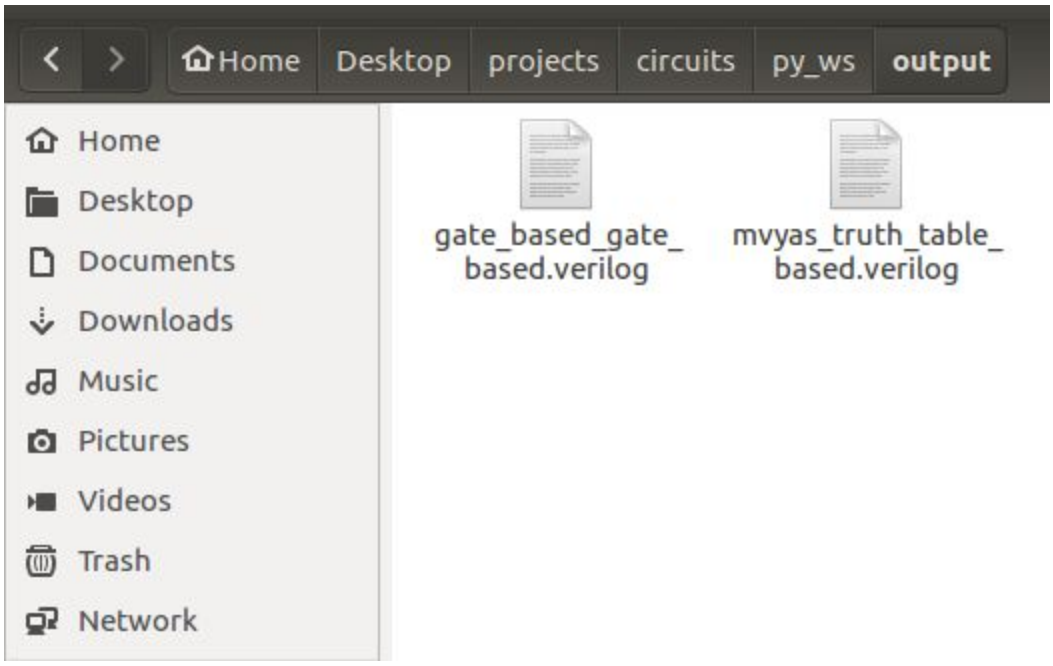
We can see the output file here.



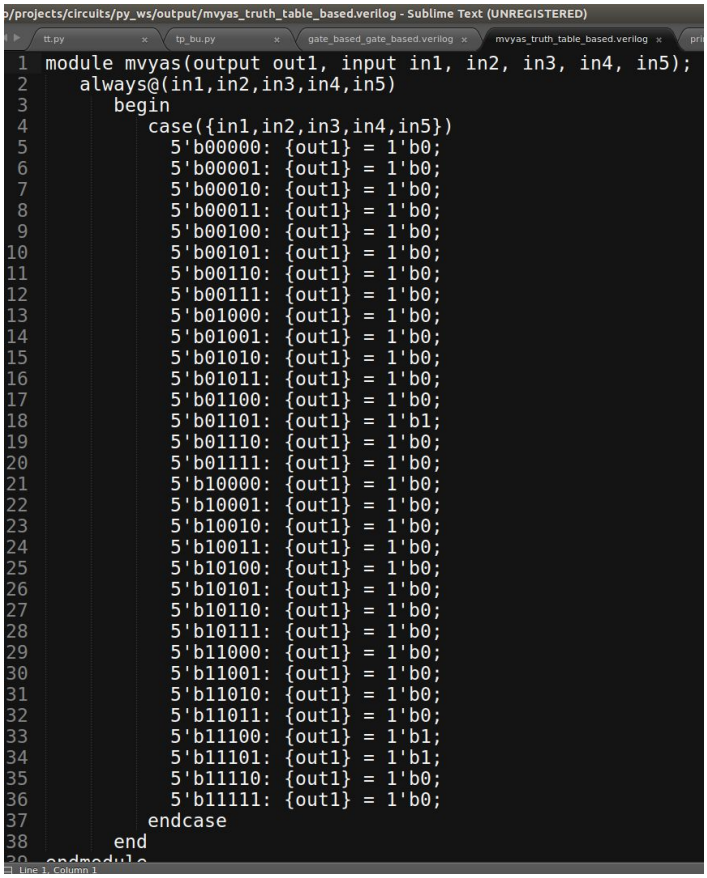
Truth table based verilog conversion.



The software generates a verilog file named gate_based.verilog in the output directory.

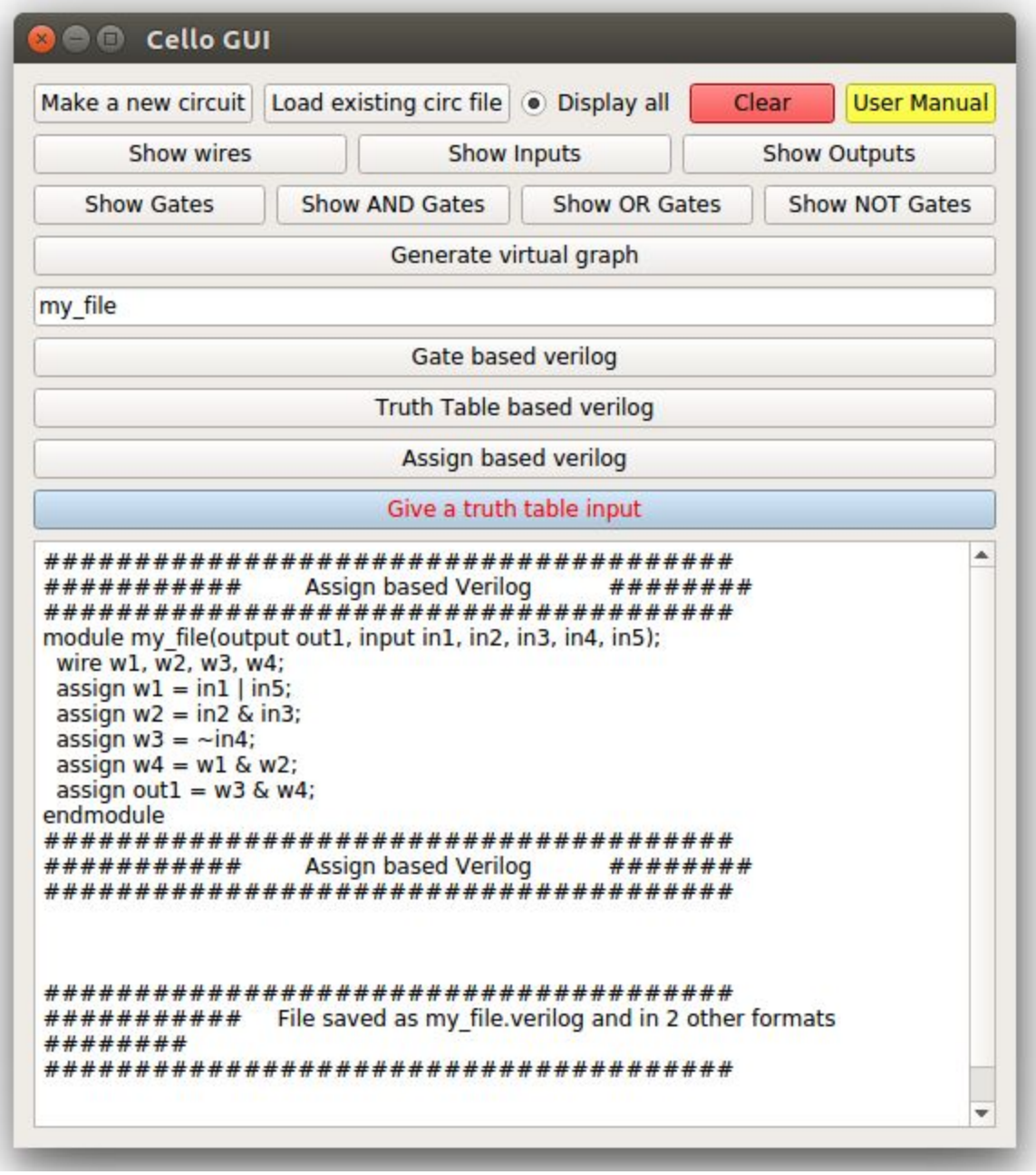


We can see the output file here.

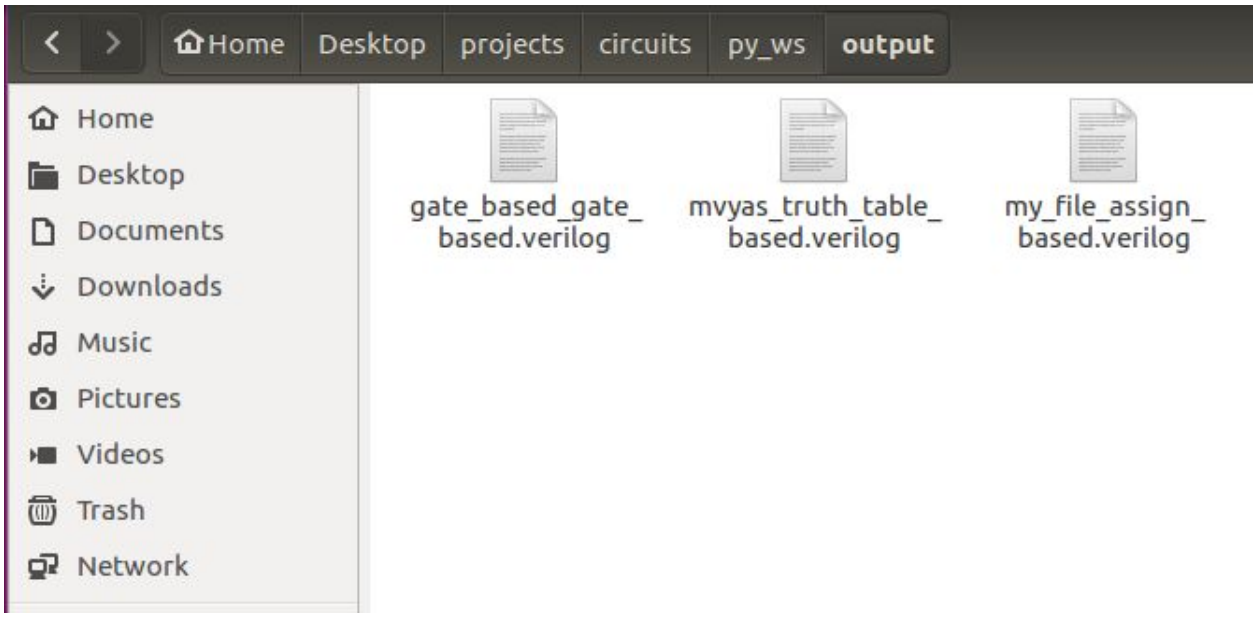



```
module mvyas(output out1, input in1, in2, in3, in4, in5);
   always@(in1,in2,in3,in4,in5)
      begin
         case({in1,in2,in3,in4,in5})
           5'b00000: {out1} = 1'b0;
           5'b00001: {out1} = 1'b0;
           5'b00010: {out1} = 1'b0;
           5'b00011: {out1} = 1'b0;
           5'b00100: {out1} = 1'b0;
           5'b00101: {out1} = 1'b0;
           5'b00110: {out1} = 1'b0;
           5'b00111: {out1} = 1'b0;
           5'b01000: {out1} = 1'b0;
           5'b01001: {out1} = 1'b0;
           5'b01010: {out1} = 1'b0;
           5'b01011: {out1} = 1'b0;
           5'b01100: {out1} = 1'b0;
           5'b01101: {out1} = 1'b1;
           5'b01110: {out1} = 1'b0;
           5'b01111: {out1} = 1'b0;
           5'b10000: {out1} = 1'b0;
           5'b10001: {out1} = 1'b0;
           5'b10010: {out1} = 1'b0;
           5'b10011: {out1} = 1'b0;
           5'b10100: {out1} = 1'b0;
           5'b10101: {out1} = 1'b0;
           5'b10110: {out1} = 1'b0;
           5'b10111: {out1} = 1'b0;
           5'b11000: {out1} = 1'b0;
           5'b11001: {out1} = 1'b0;
           5'b11010: {out1} = 1'b0;
           5'b11011: {out1} = 1'b0;
           5'b11100: {out1} = 1'b1;
           5'b11101: {out1} = 1'b1;
           5'b11110: {out1} = 1'b0;
           5'b11111: {out1} = 1'b0;
         endcase
      end
endmodule
```

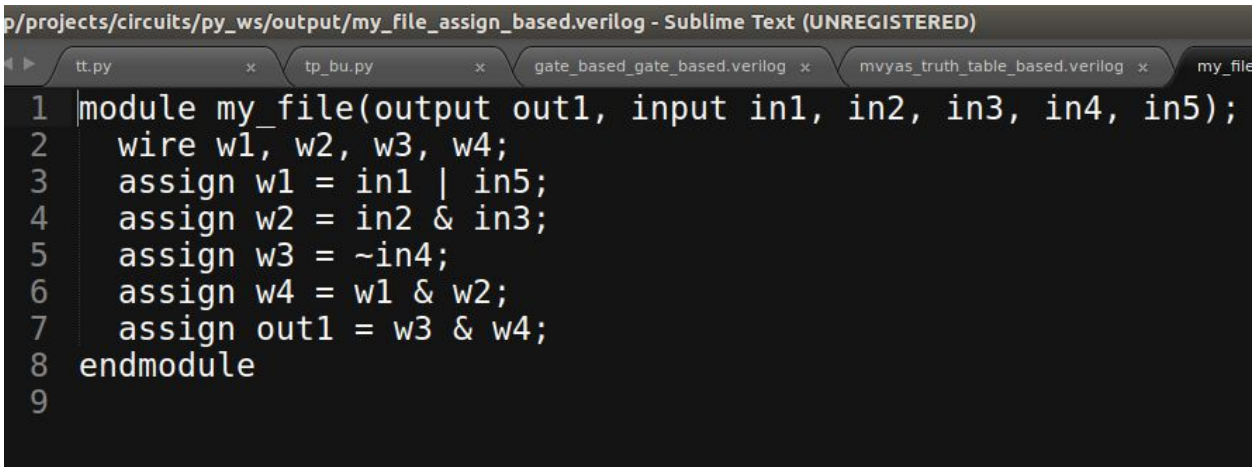
Assign based verilog conversion.



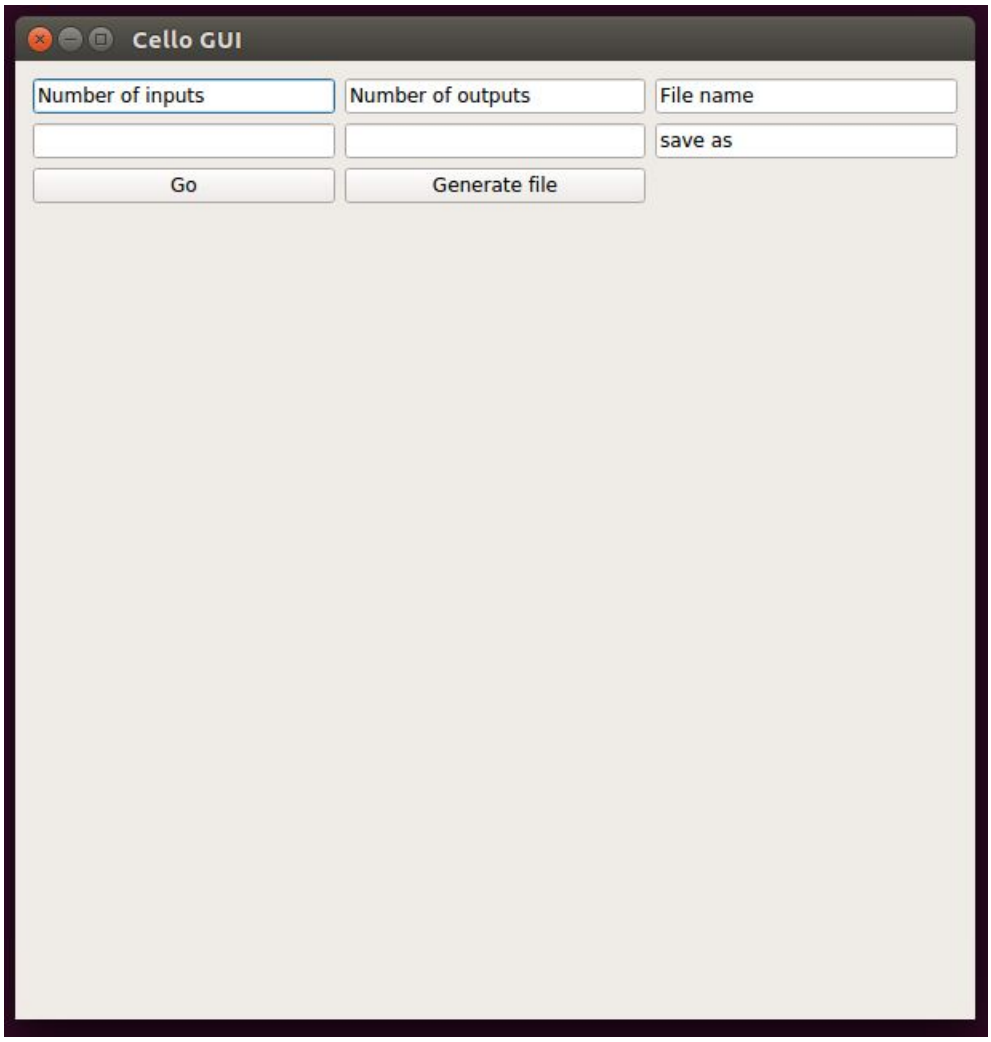
The software generates a verilog file named gate_based.verilog in the output directory.



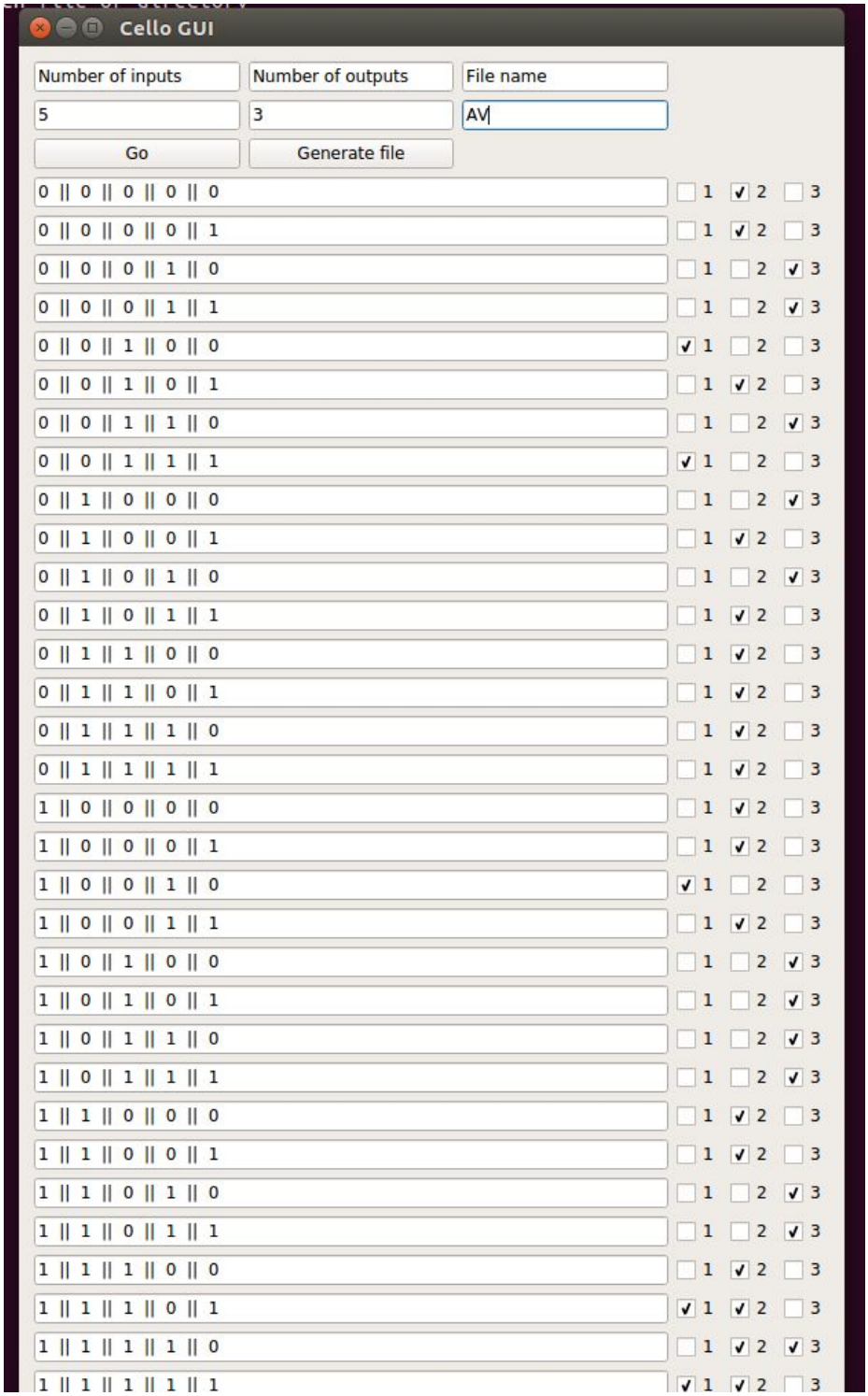
We can see the output file here.



Edit an truth table based file graphically.



Truth table maker.



Output:

```
module AV(output out1, output out2, output out3, input in1, in2, in3, in4, in5);
   always@(in1,in2,in3,in4,in5)
      begin
         case({in1,in2,in3,in4,in5})
           5'b00000: {out1, out2, out3} = 3'b010;
           5'b00001: {out1, out2, out3} = 3'b010;
           5'b00010: {out1, out2, out3} = 3'b001;
           5'b00011: {out1, out2, out3} = 3'b001;
           5'b00100: {out1, out2, out3} = 3'b100;
           5'b00101: {out1, out2, out3} = 3'b010;
           5'b00110: {out1, out2, out3} = 3'b001;
           5'b00111: {out1, out2, out3} = 3'b100;
           5'b01000: {out1, out2, out3} = 3'b001;
           5'b01001: {out1, out2, out3} = 3'b010;
           5'b01010: {out1, out2, out3} = 3'b001;
           5'b01011: {out1, out2, out3} = 3'b010;
           5'b01100: {out1, out2, out3} = 3'b010;
           5'b01101: {out1, out2, out3} = 3'b010;
           5'b01110: {out1, out2, out3} = 3'b010;
           5'b01111: {out1, out2, out3} = 3'b010;
           5'b10000: {out1, out2, out3} = 3'b010;
           5'b10001: {out1, out2, out3} = 3'b010;
           5'b10010: {out1, out2, out3} = 3'b100;
           5'b10011: {out1, out2, out3} = 3'b010;
           5'b10100: {out1, out2, out3} = 3'b001;
           5'b10101: {out1, out2, out3} = 3'b001;
           5'b10110: {out1, out2, out3} = 3'b001;
           5'b10111: {out1, out2, out3} = 3'b001;
           5'b11000: {out1, out2, out3} = 3'b010;
           5'b11001: {out1, out2, out3} = 3'b010;
           5'b11010: {out1, out2, out3} = 3'b001;
           5'b11011: {out1, out2, out3} = 3'b001;
           5'b11100: {out1, out2, out3} = 3'b010;
           5'b11101: {out1, out2, out3} = 3'b110;
           5'b11110: {out1, out2, out3} = 3'b011;
           5'b11111: {out1, out2, out3} = 3'b110;
         endcase
      end
endmodule
```

Legal Stuff:

Licence

Cello GUI
Licence for use and Distribution

Documentation version 1.0

Credits & Team

Principal Investigator: Dr. Manish K. Gupta
Graduate Mentor: Dixita Limbachiya
Developers: Vandit Modi, Smit Prajapati, Dhruv Patel, Vaibhav Amit Patel, Dhaval Parmar, Maharshi Vyas.

Note: You can use Cello GUI on any computer, including a computer in a commercial organization. You don't need to register or pay for Cello GUI.

Warranty

Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

References and Feedback

Users are requested to contact team at the feedback page on the website www.guptalab.org for any issue with the software or they can also mail at not_yet_made@guptalab.org. A cross platform python script to run the GUI is available on the project home page along with source code.