BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Electrical and Electronic Engineering

Name of the Project: Digital Logic Circuit Implementation

Course No. : EEE 212

Course Title: Numerical Technique Laboratory

Submitted by,

Name: Abdul Jabber Antor   ID: 1906062

Section: A2

Level : 2   **Term :** 1

Date of Submission : 14 -02 -2022

Introduction:

In this project, MATLAB is used to build a program which can-

- Generate truth table based on a given number of logical inputs.
- Generate logical output function from the truth table
- Generate logic circuit from the function.

App Designer was used as Graphical User Interface (GUI) in this project.

How to Run:

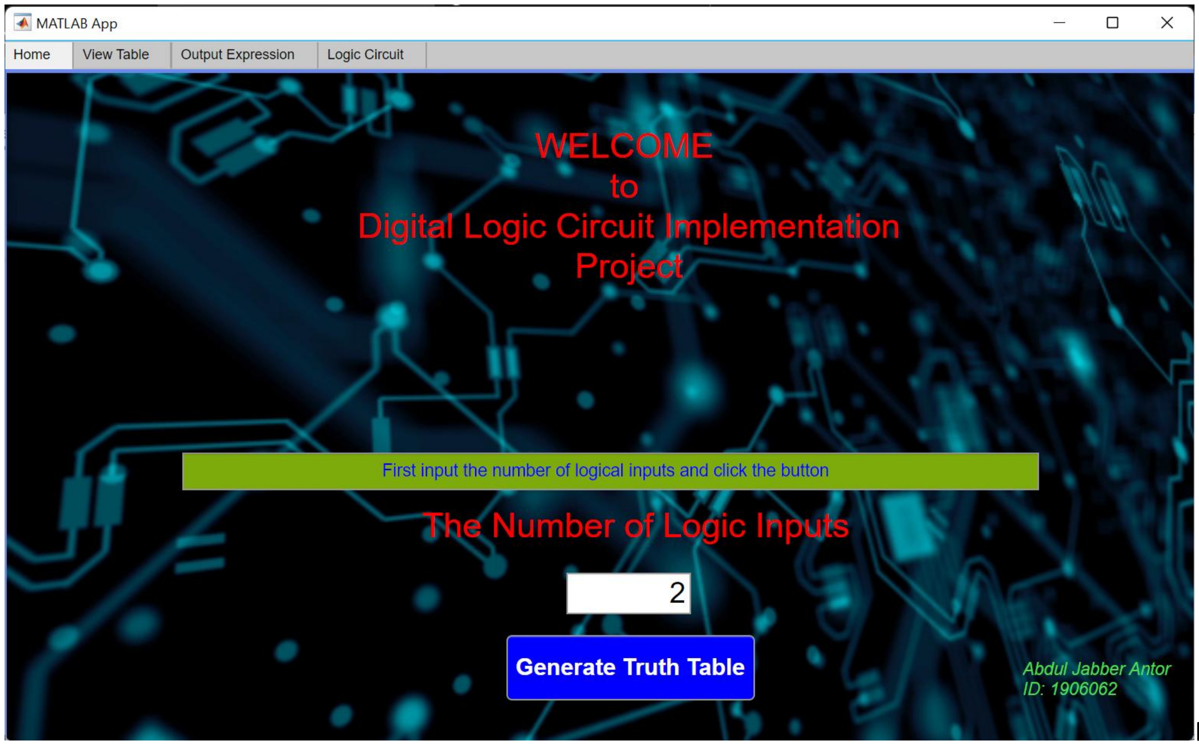
1. Download/clone this repository.

2. Open `LogicDiagram.mlapp` in MATLAB App Designer.

3. Click **Run**.

Interface:

After running the .mlapp file, a window containing four tabs will be visible to the user. The four tabs are named- Home, View Table, Output Expression and Logic Circuit which is shown below-



WELCOME
to
Digital Logic Circuit Implementation Project

First input the number of logical inputs and click the button

The Number of Logic Inputs

2

Abdul Jabber Antor
ID: 1906062

Fig: Initial interface

Home:

The first interface which will be visible is the home tab.

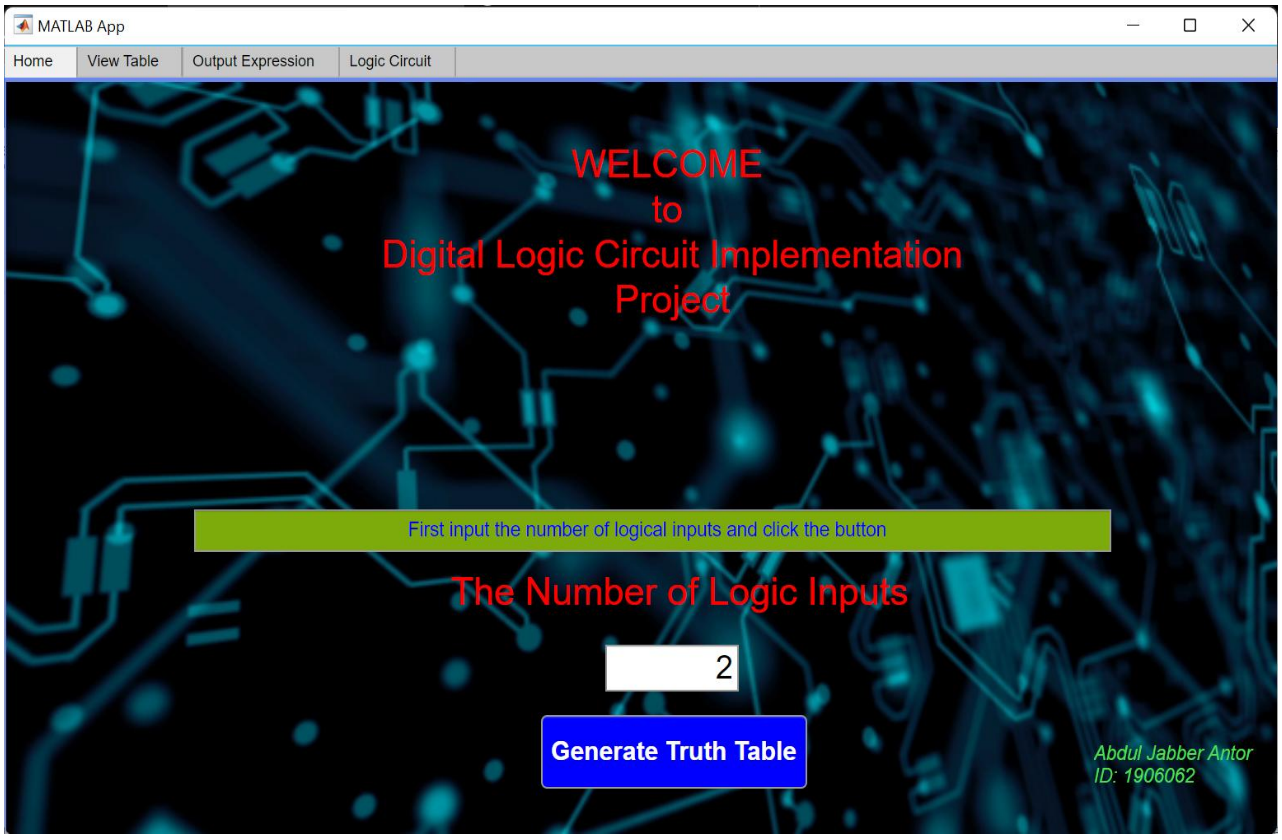

MATLAB App
Home
View Table
Output Expression
Logic Circuit
WELCOME
to
Digital Logic Circuit Implementation
Project
First input the number of logical inputs and click the button
The Number of Logic Inputs
2
Generate Truth Table
Abdul Jabber Antor
ID: 1906062

Fig: Interface of Home Tab.

Here, the user will find an edit box to enter the number of inputs and a button to generate a truth table based on the given inputs. An instruction “First input the number of logical inputs and click the button” will be visible so that the user can understand what to do. The default value for the input was set 2. After clicking the “Generate Truth Table” button, “View Table” tab will be displayed.

View Table:

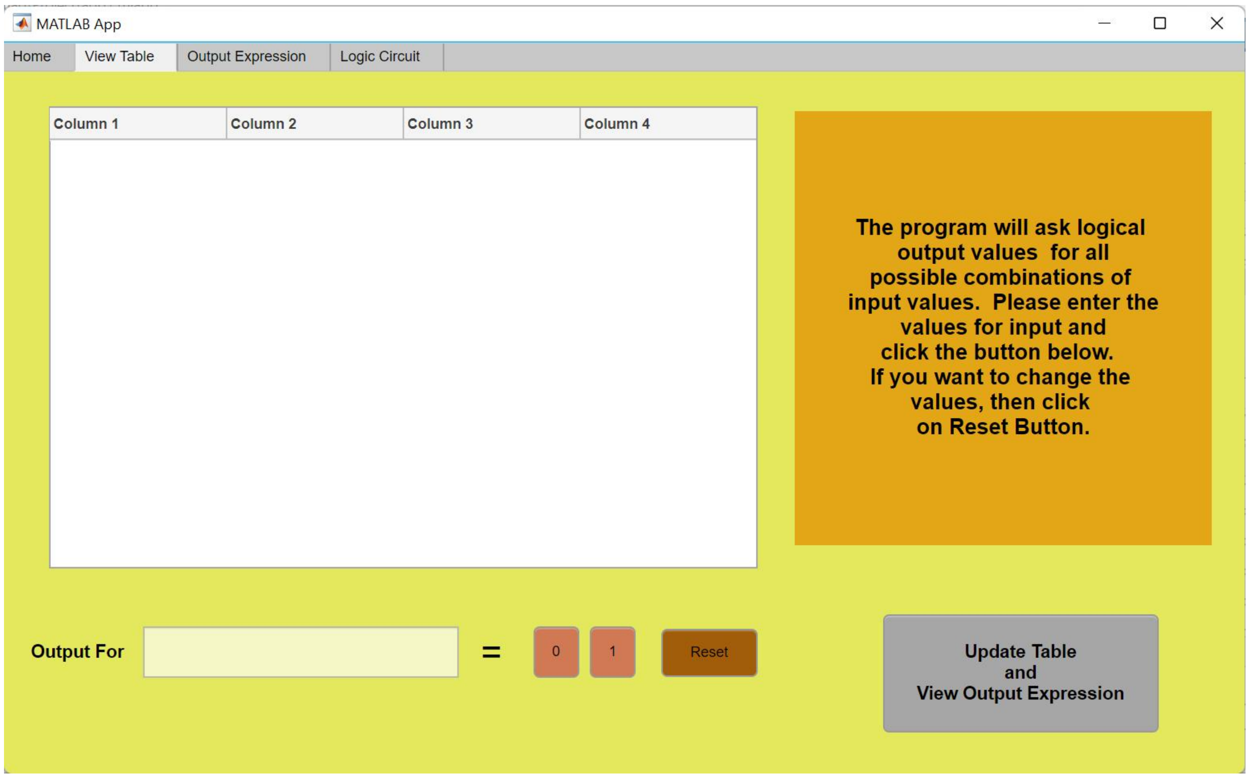

The program will ask logical output values for all possible combinations of input values. Please enter the values for input and click the button below. If you want to change the values, then click on Reset Button.

Fig: Interface of View Table Tab.

Here, a table is situated on the right side of the screen which will contain the truth table. Below the table, the program will ask for output value for all combinations of input. The user can choose either 0 or 1 as output. Instructions are also displayed on the right side of the window describing the procedures. After completing the table, the user has to click the button on the right-down part of the screen to see the output expression. Then the “Output Expression” tab will be opened. If the user feels to change the inputs, he/she can click the “Reset” button. Then the user will return to the “Home” tab.

Output Expression:

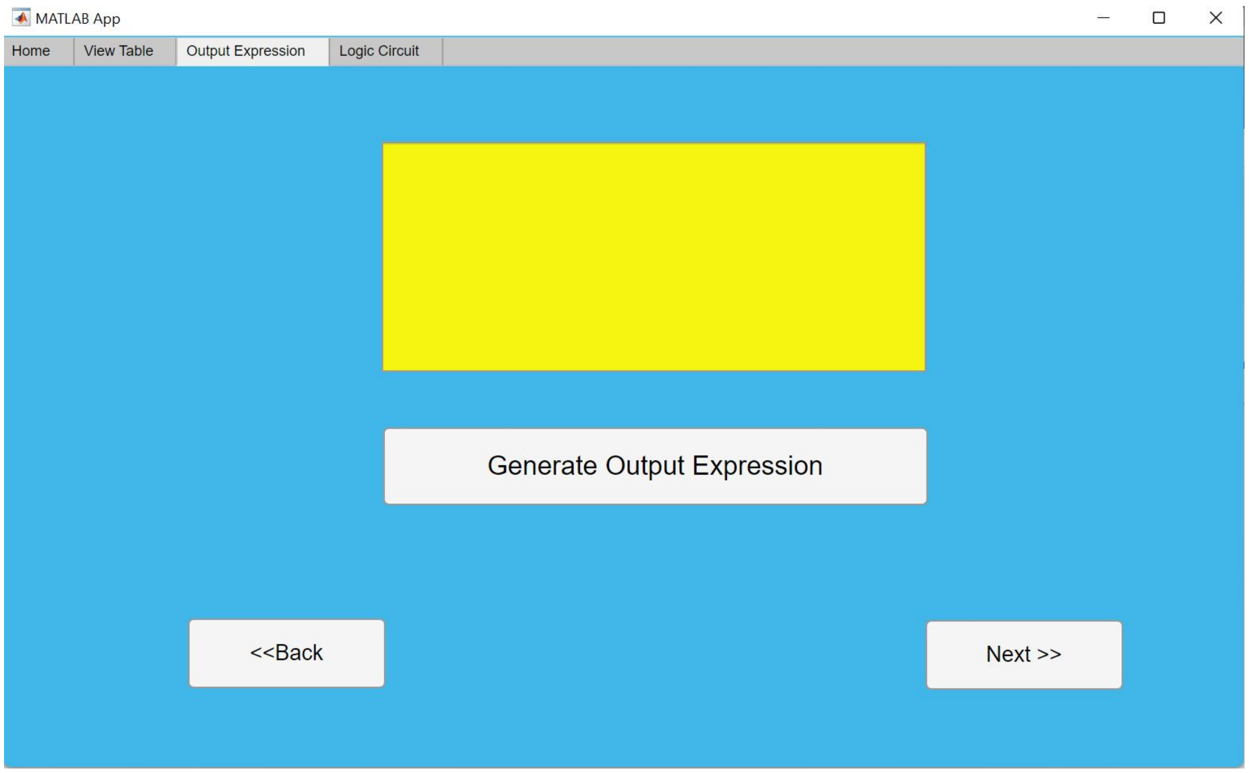
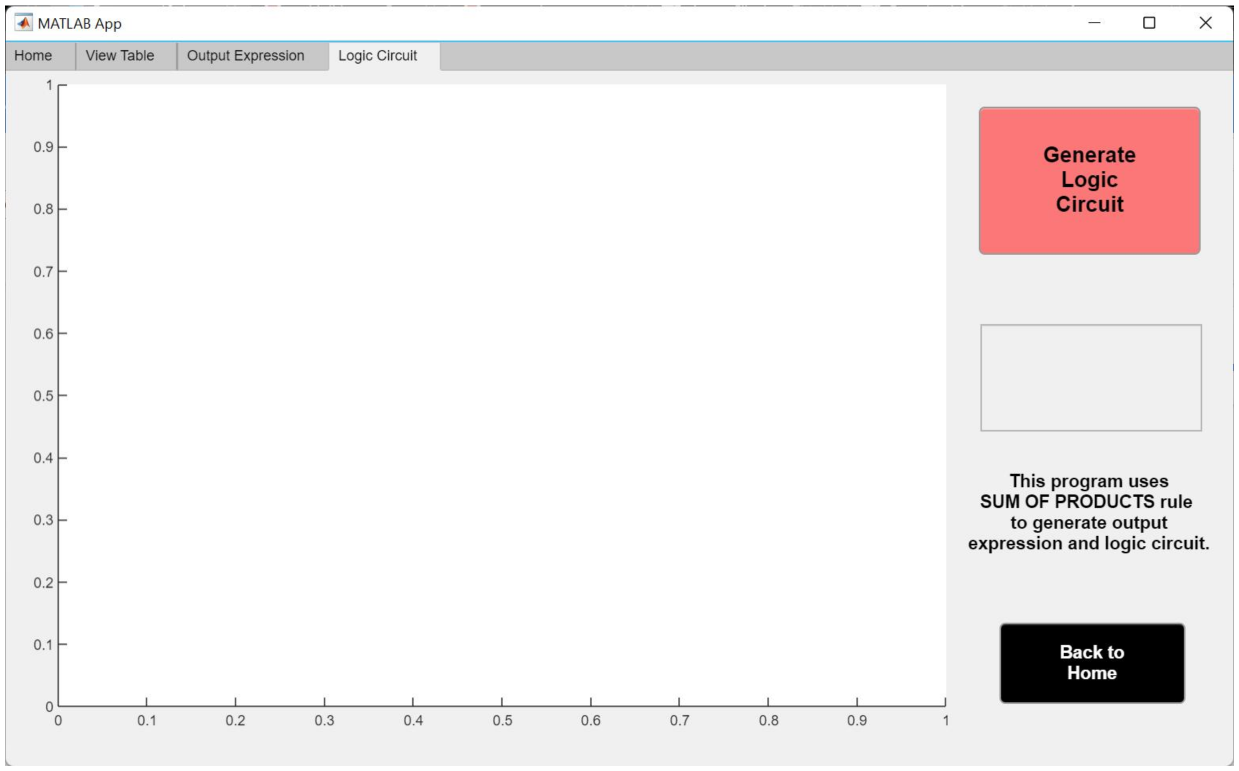


Fig: Interface of Output Expression Tab.

In this tab, there is one display box and three buttons. Clicking the “Generate Output Expression” button will generate the Boolean output expression as the name suggests. The program uses SUM OF PRODUCTS rule to generate output expression. Clicking the “Back” button will cause the “View Table” tab to open and clicking the “Next” button will cause the “Logic circuit” tab to open. However, the user can move between the tabs by simply clicking on them.

Logic Circuit:



This program uses SUM OF PRODUCTS rule to generate output expression and logic circuit.

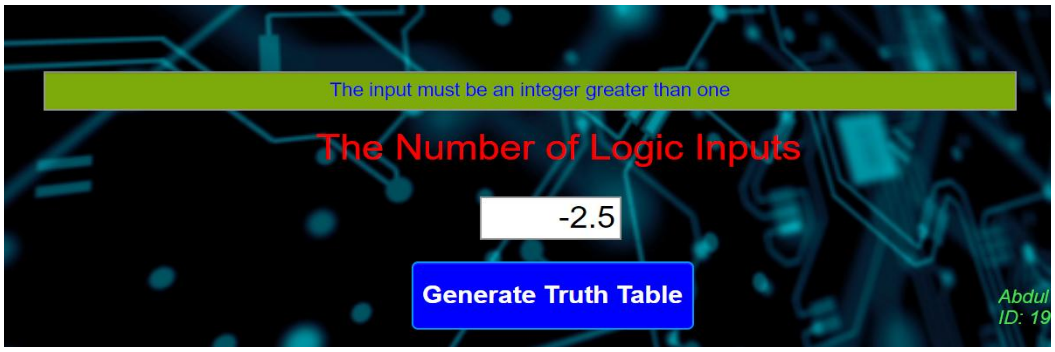
Fig: Interface of Logic Circuit Tab.

The main portion of this tab is a plotting axis which will hold the drawing of the generated logic circuit. The user has to click on the red button to generate the circuit. There is a text box below the button which will display relevant message to the output circuit. Again, there is a “Back to Home” button which will take the user to the “Home” tab.

Now, the working of the project for 10 different cases will be discussed.

Results for different cases:

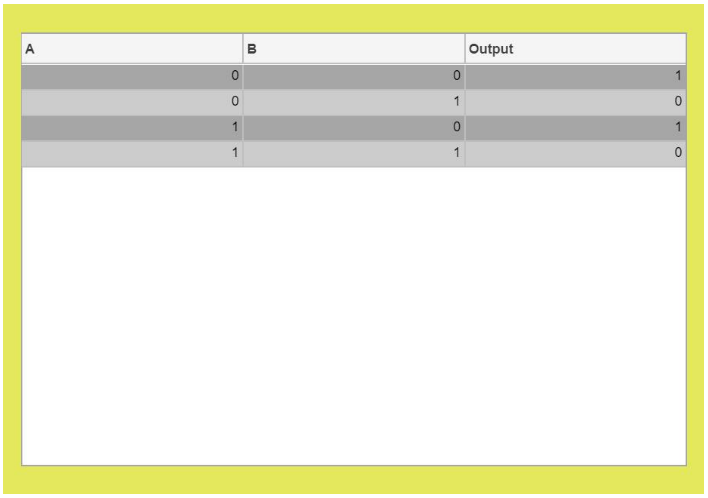
Case 1: Let, the user has given some negative or absurd value as input.


The input must be an integer greater than one

After clicking on the "Generate Truth Table" button, a prompt message will be shown "The input must be an integer greater than one" and the app will not run until the user gives proper value as input.

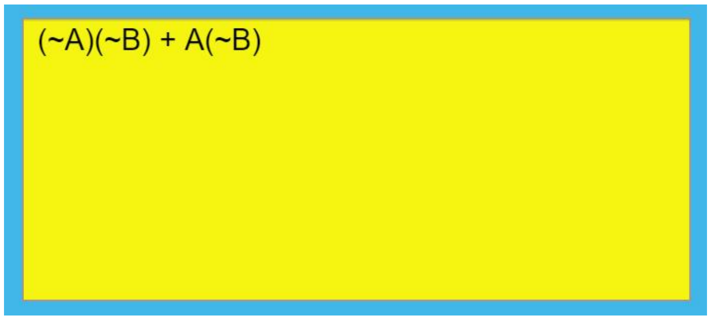
Case 2: Number of inputs = 2, some random outputs.

Truth Table:



A	B	Output
0	0	1
0	1	0
1	0	1
1	1	0

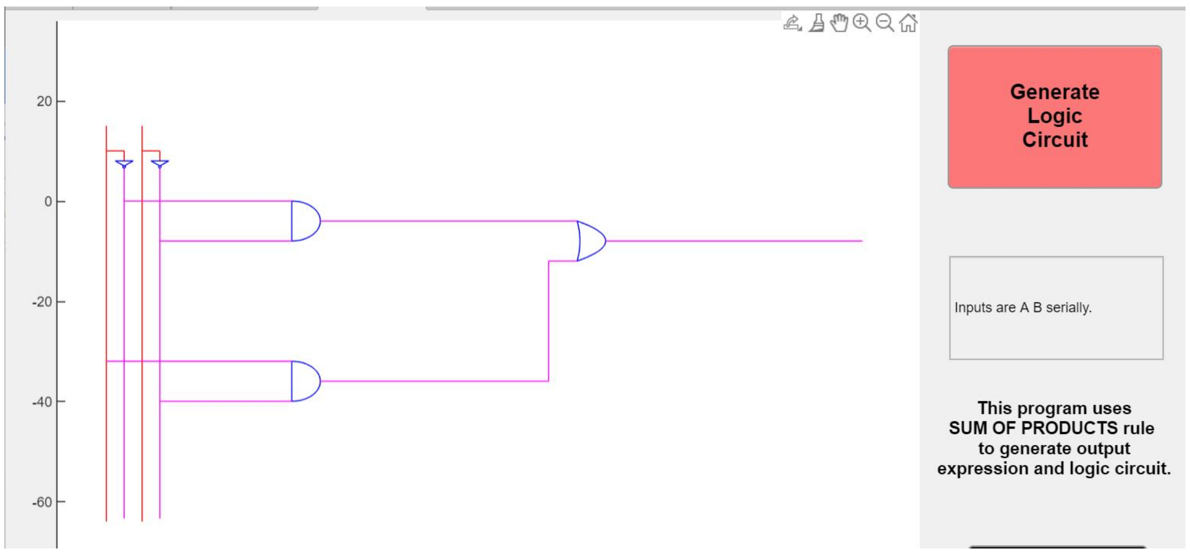
Output Expression:



(~A)(~B) + A(~B)

Here, "~" sign represents NOT operation

Logic Circuit:



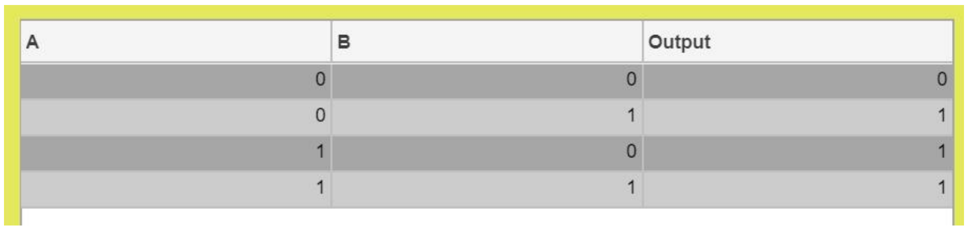
Inputs are A B serially.

This program uses SUM OF PRODUCTS rule to generate output expression and logic circuit.

The dialogue box below the push button describes that the first input line is for A and the second input line is for B.

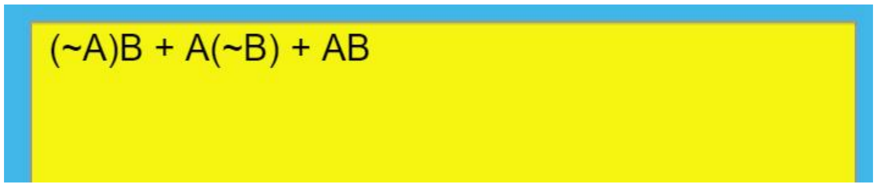
Case 3: Number of inputs = 2, another set of random outputs.

Truth Table:



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

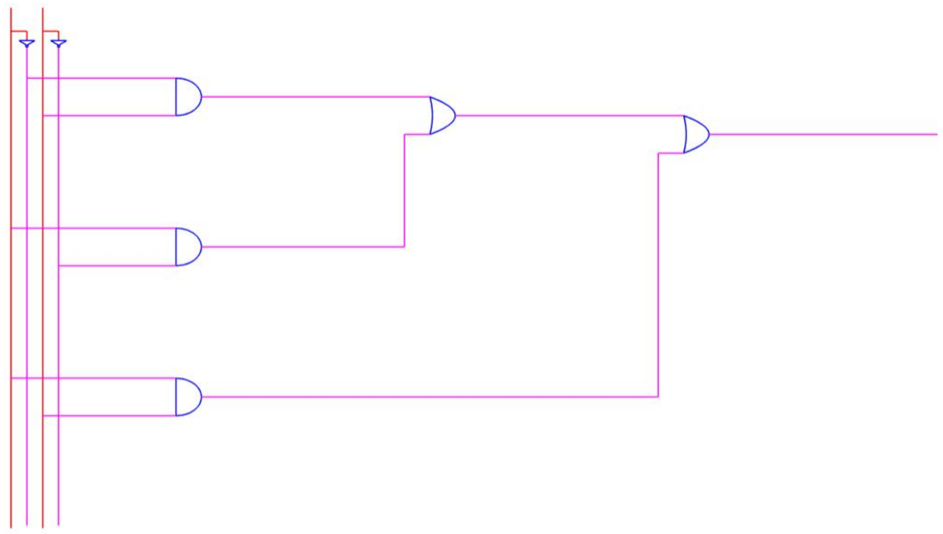
Output Expression:



(~A)B + A(~B) + AB

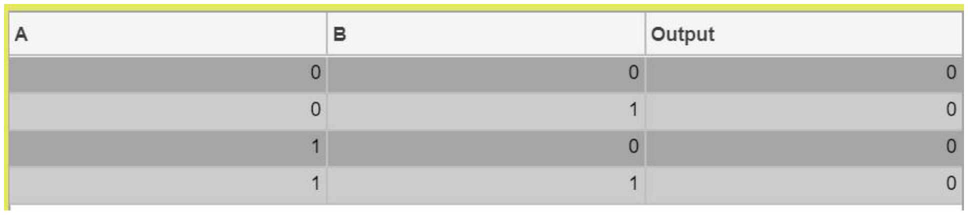
Here, "~" sign represents NOT operation

Logic Circuit:



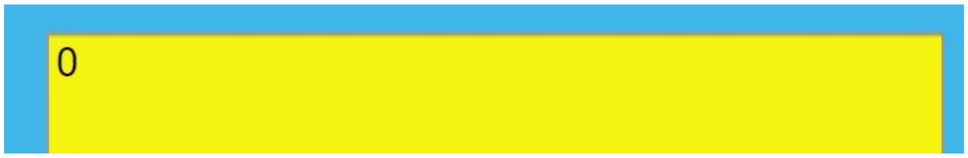
Case 4: Number of inputs = 2, all outputs are 0.

Truth Table:



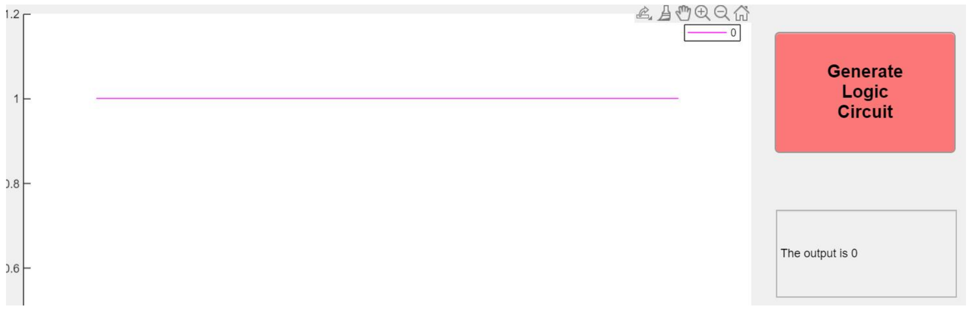
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	0

Output Expression:



0

Logic Circuit:

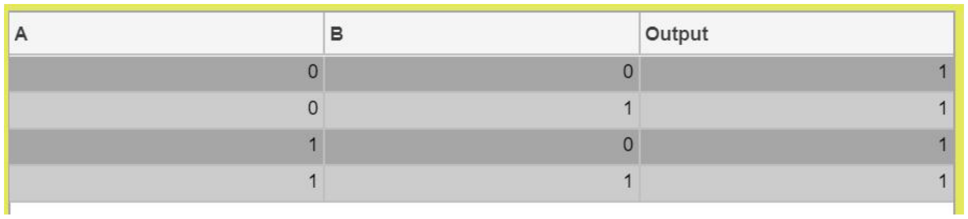


The output is 0

In this case, the output is 0, and the plot shows only a straight line with a legend “0”. The dialogue box below the button also specifies the output.

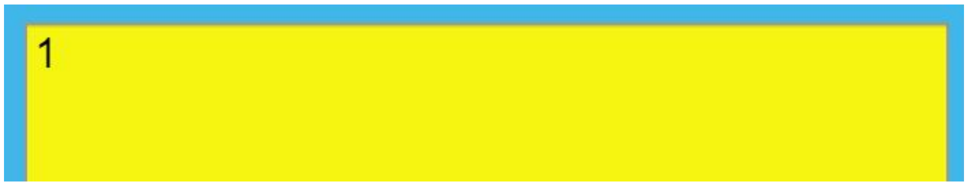
Case 5: Number of inputs = 2, all outputs are 1.

Truth Table:



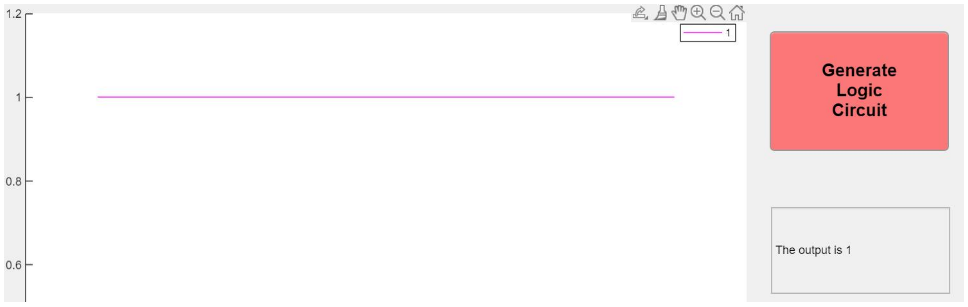
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	1

Output Expression:



1

Logic Circuit:



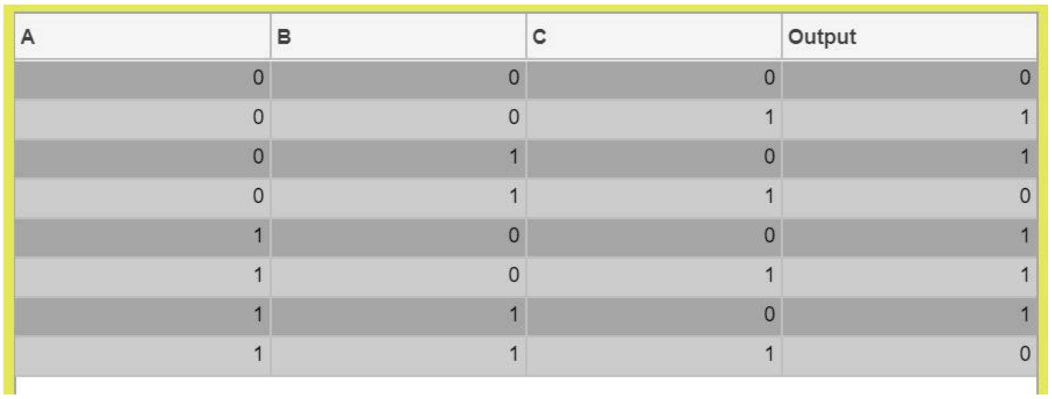
The output is 1

In this case, the output is 1, and the plot shows only a straight line

with a legend “1”. The dialogue box below the button also specifies the output.

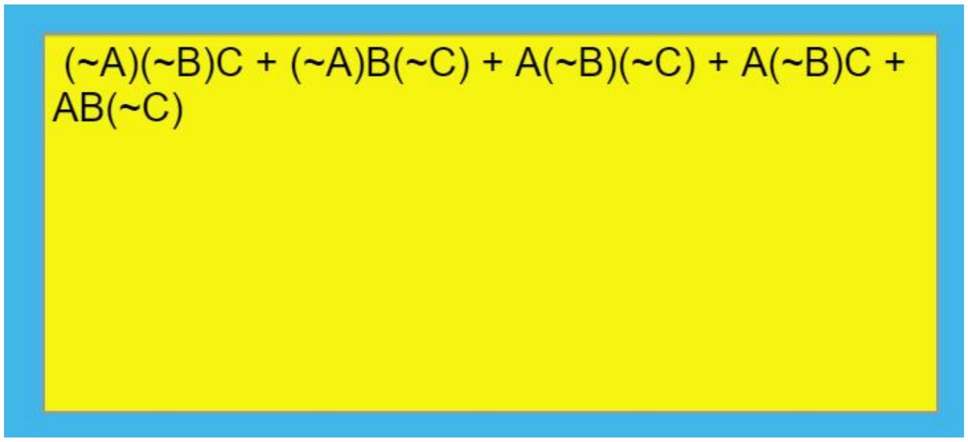
Case 6: Number of inputs =3, some random outputs.

Truth Table:



A	B	C	Output
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

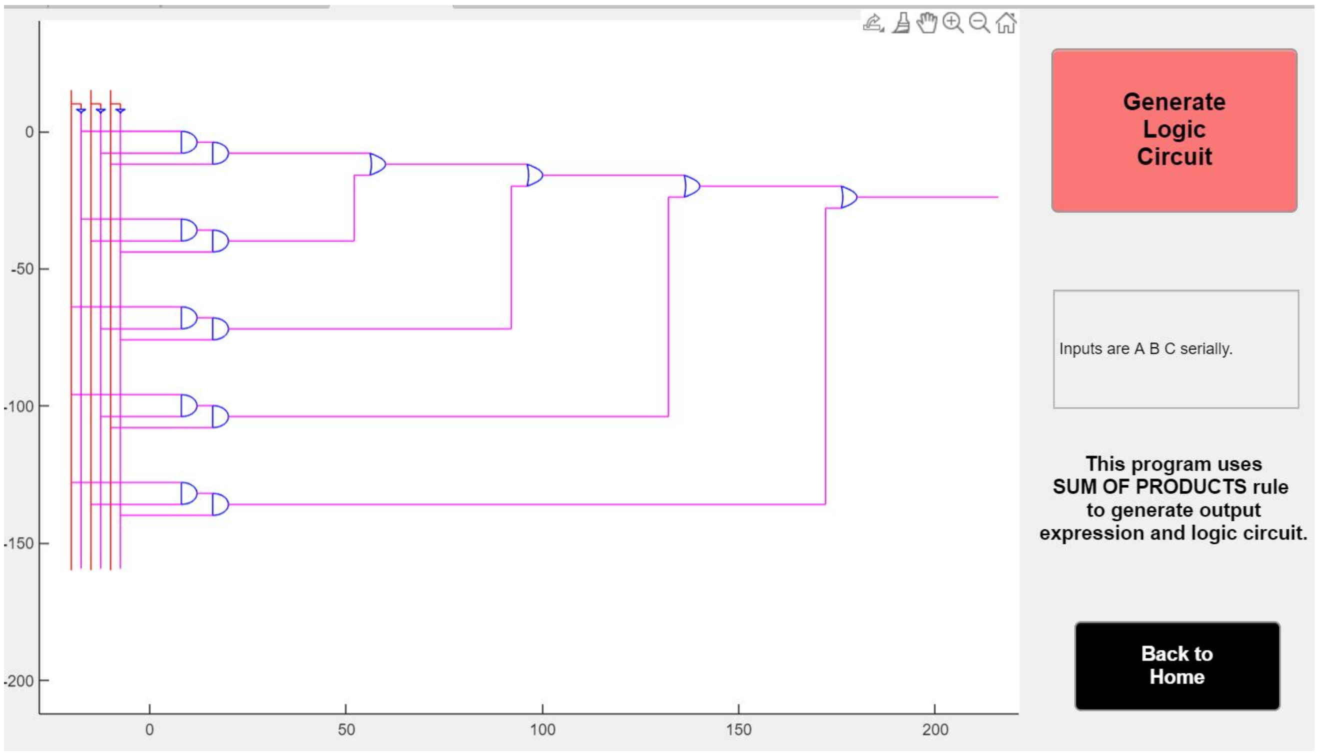
Output Expression:



(~A)(~B)C + (~A)B(~C) + A(~B)(~C) + A(~B)C + AB(~C)

Here, "~" sign represents NOT operation

Logic Circuit:



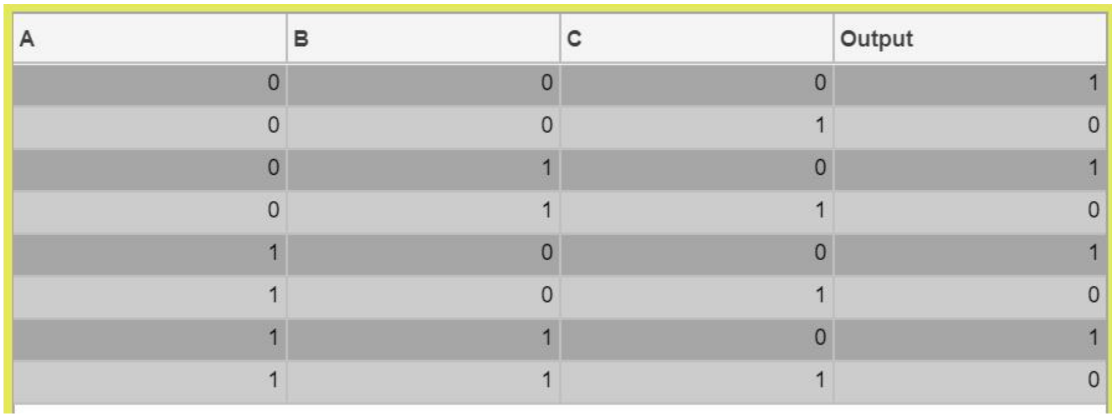
Inputs are A B C serially.

This program uses SUM OF PRODUCTS rule to generate output expression and logic circuit.

The dialogue box below the push button describes that the first input line is for A, the second input line is for B and the third input is for C.

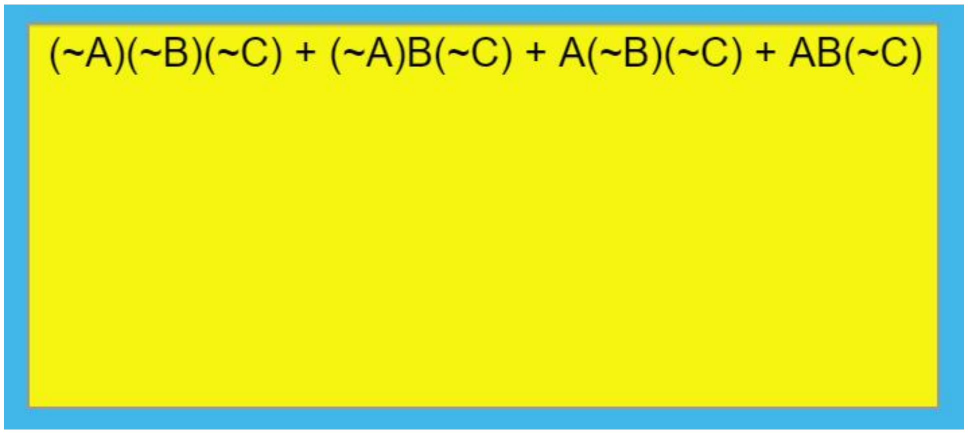
Case 7: Number of inputs = 3, another set of random outputs.

Truth Table:



A	B	C	Output
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	1
1	1	1	0

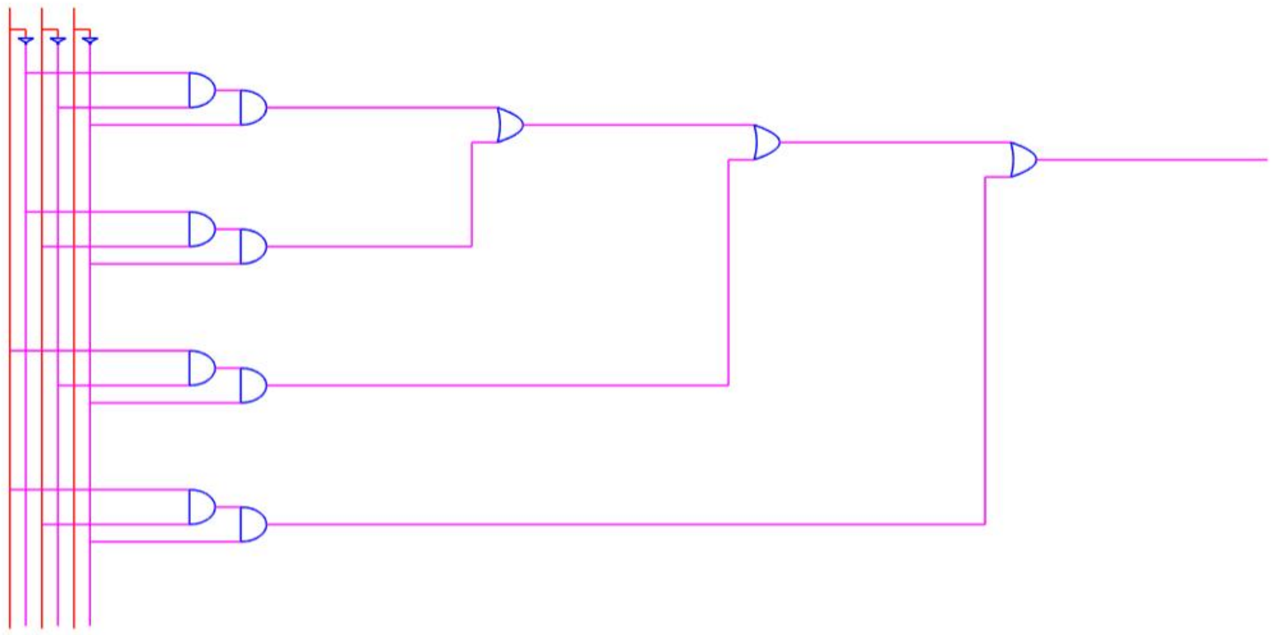
Output Expression:



(~A)(~B)(~C) + (~A)B(~C) + A(~B)(~C) + AB(~C)

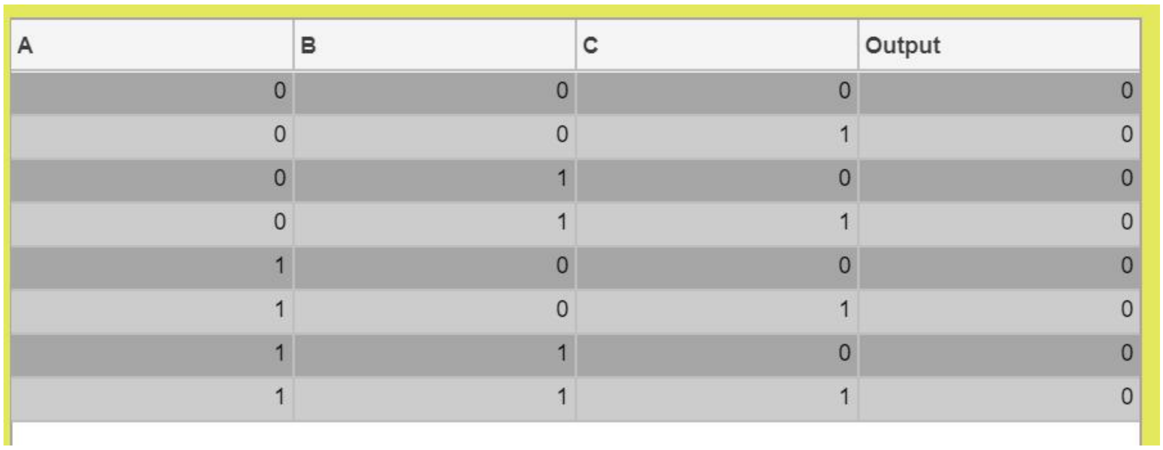
Here, "~" sign represents NOT operation

Logic Circuit:



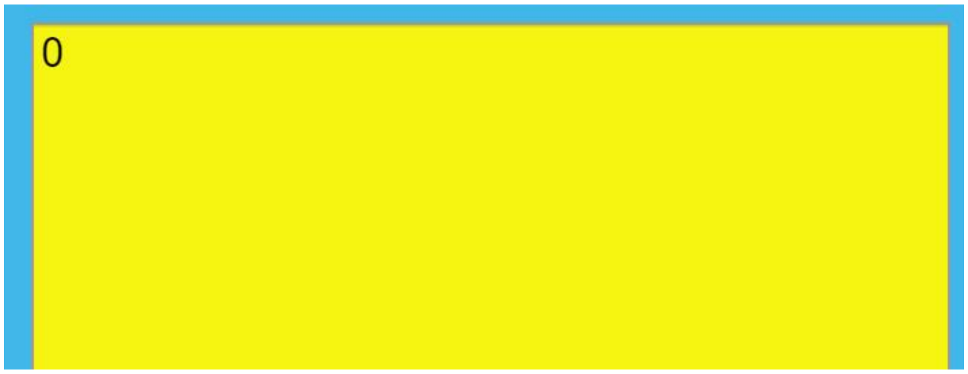
Case 8: Number of inputs = 3, all outputs are 0.

Truth Table:



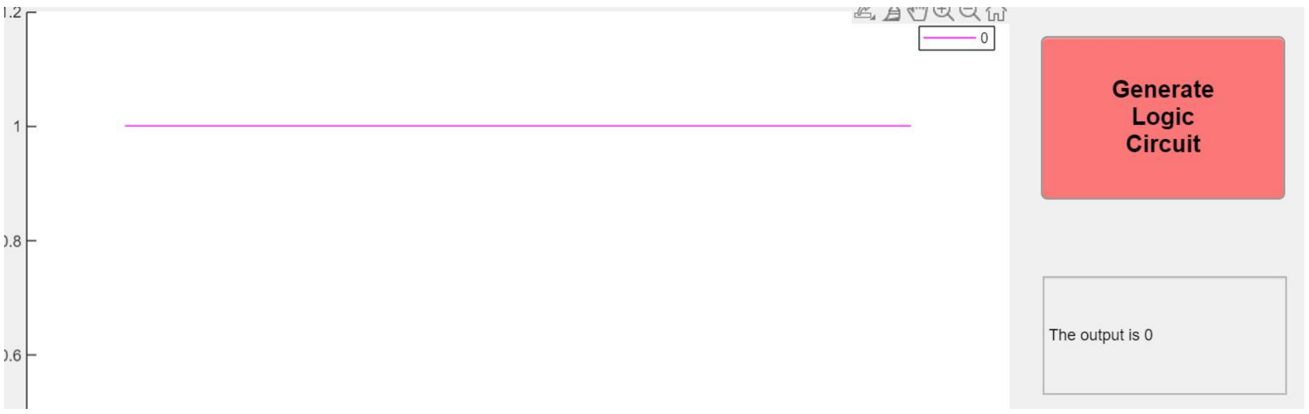
A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Output Expression:



0

Logic Circuit:



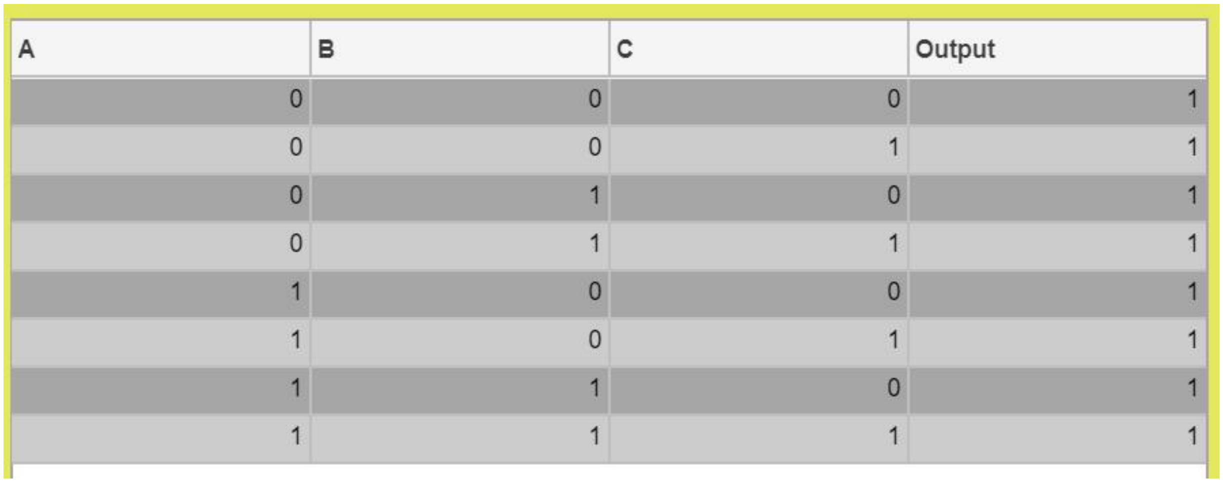
The output is 0

In this case, the output is 0, and the plot shows only a straight line

with a legend “0”. The dialogue box below the button also specifies the output.

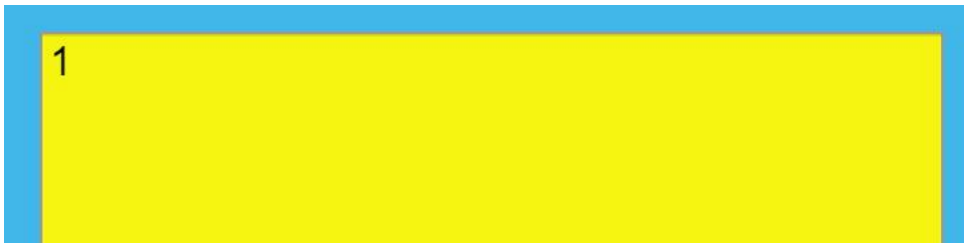
Case 9: Number of inputs = 3, all outputs are 1.

Truth Table:



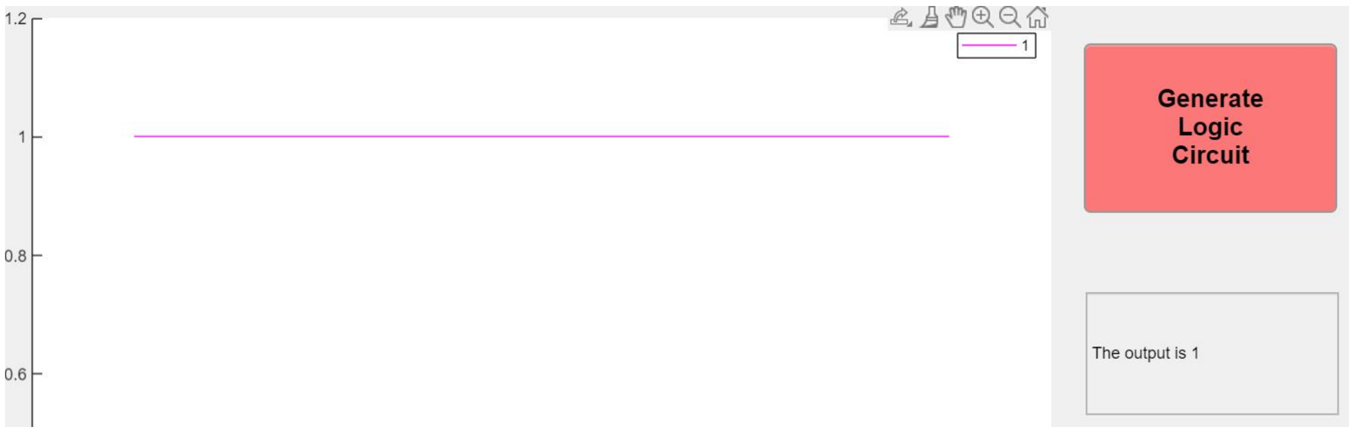
A	B	C	Output
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Output Expression:



1

Logic Circuit:



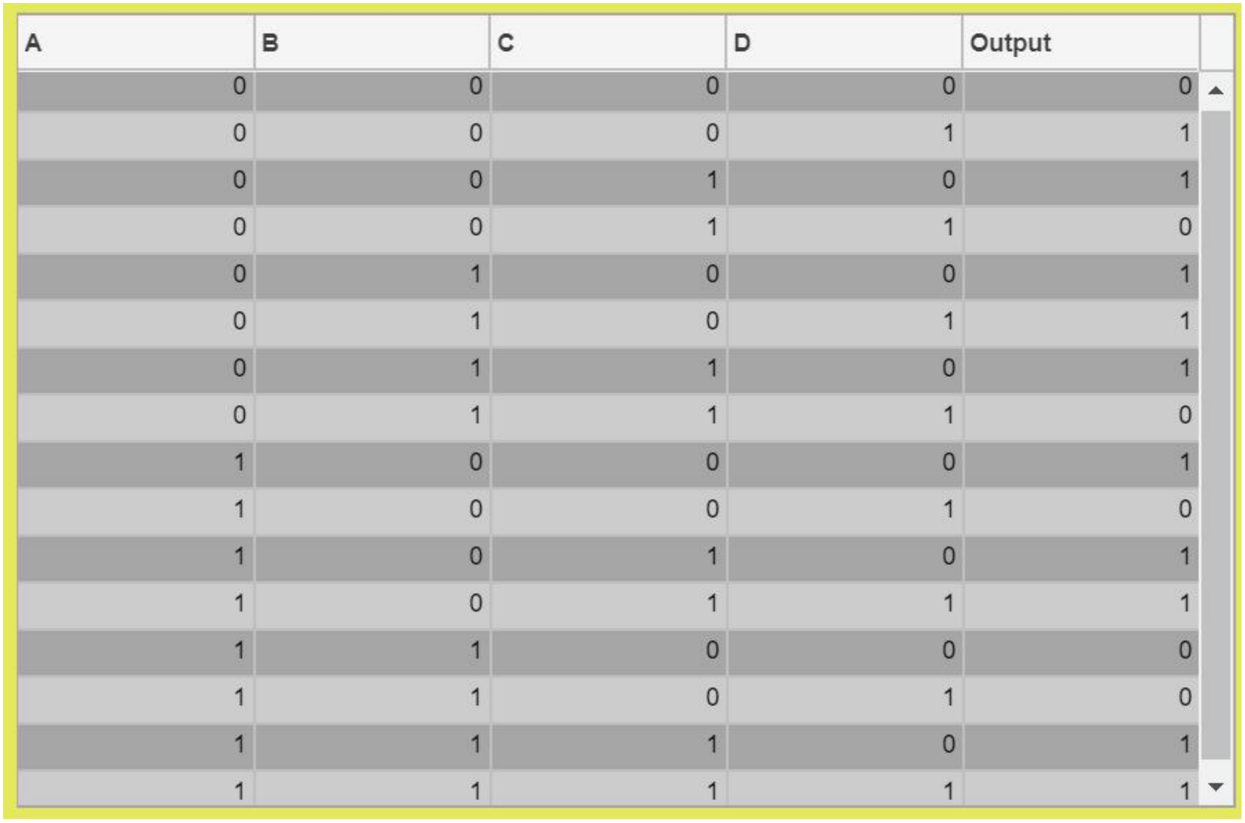
The output is 1

In this case, the output is 0, and the plot shows only a straight line

with a legend “0”. The dialogue box below the button also specifies the output.

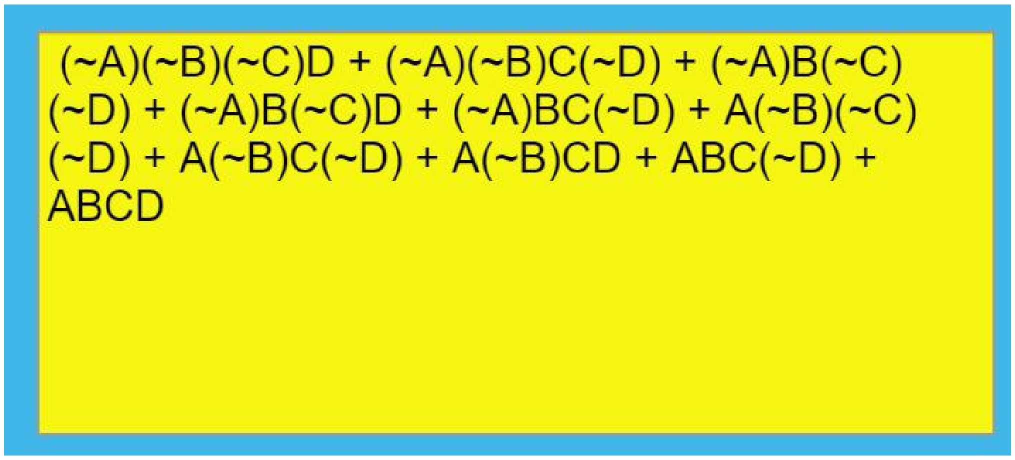
Case 10: Number of inputs = 4, some random outputs.

Truth Table:



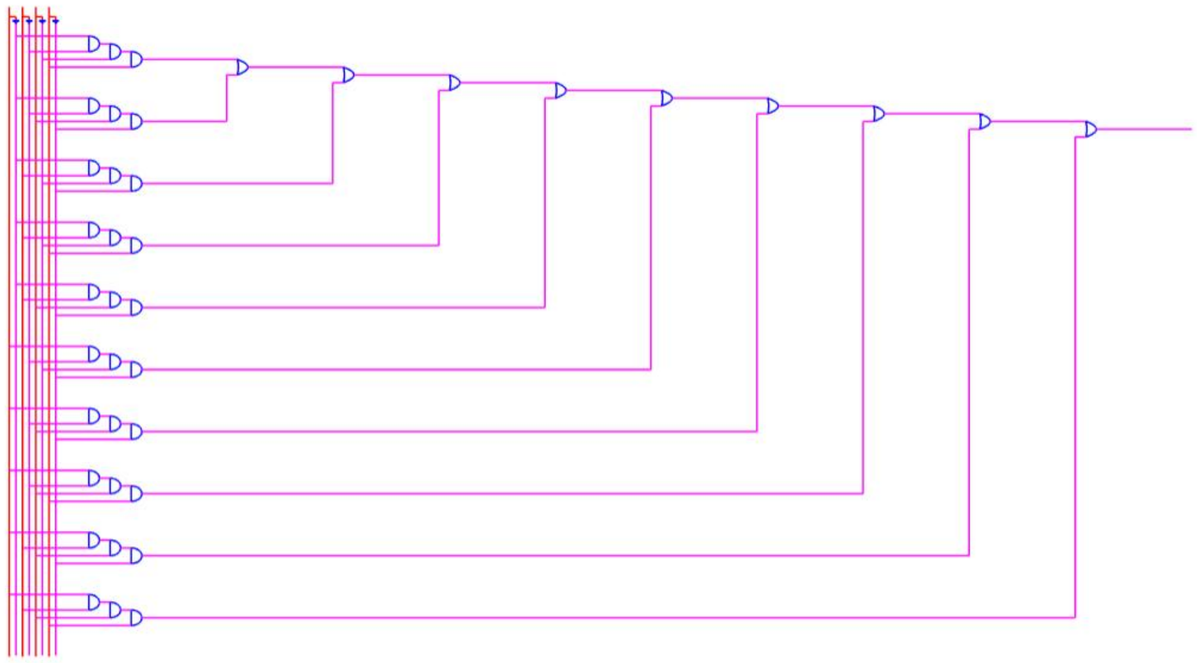
A	B	C	D	Output
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Output Expression:



(~A)(~B)(~C)D + (~A)(~B)C(~D) + (~A)B(~C)(~D) + (~A)B(~C)D + (~A)BC(~D) + A(~B)(~C)(~D) + A(~B)C(~D) + A(~B)CD + ABC(~D) + ABCD

Logic Circuit:



MATLAB Code:

```
  properties (Access = public)

    n        %number of inputs

    B=1       %Button click counter

    m        %Matrix for truth table

    inpDone=0;  %Checks if the input is correct or not

  end




% Callbacks that handle component events

  methods (Access = private)




    % Button pushed function: GenerateTruthTableButton

    function GenerateTruthTableButtonPushed(app, event)

      app.B=1; %initialize Button Counter




      %Asking for first input for all possible combinations

      app.OutputForEditField.Value =  num2str(zeros(1,app.n));


      %Initializes for next use

      app.OutputExp.Value= " ";

      app.plotguide.Value= " ";

      hold(app.graph,'off')
```

```
plot(app.graph,1,1);

hold(app.graph,'off')

app.UITable.Data=0;


%Reads the number of inputs

app.n=app.N.Value;



%Generating truth table

if app.n>=2 && fix(app.n)-app.n == 0

rows=2^app.n;

cols=app.n;

app.m=ones(rows,cols);

    for i = 1:cols

        j=1;

        while j<=rows

            app.m(j,i)=0;

                if mod(j,2^(cols-i))==0

                    j=j+2^(cols-i);

                end

            j=j+1;

        end

    end



%Adding the column for output

app.m = [app.m zeros(rows,1)];
```

```
%loading the matrix in GUI

app.UITable.Data=app.m;



%Naming the columns

for i=1:cols

   app.UITable.ColumnName{i}=char(64+i);

end

app.UITable.ColumnName{i+1}="Output";





%Asking for input for all possible combinations

app.OutputForEditField.Value = num2str(zeros(1,cols));



%Guide for user

app.guide.Value="The truth table is generated. Please " + ...

   "click on ""View Table"" tab to see the truth table";



%Goes to different screen

app.TabGroup.SelectedTab = app.ViewTable;



%input taken properly

app.inpDone=1;
```

```
    else    %if input is not taken properly
        app.guide.Value= "The input must be an integer" + ...
            " greater than one";
    end




end


% Button pushed function: Button_2
function Button_2Pushed(app, event)

   if app.inpDone



        %If taking input is complete,
        % then button press will have no effect
        if app.B~=(2^app.n)+1
                app.m(app.B,end)=0;
        end
        %If taking input is complete,
        % then button press will not change the input message
        if app.B<2^(app.n)
                app.OutputForEditField.Value = num2str(app.m(app.B+1,1:end-1));
        end



        %Taking inputs for button press
        if  app.B<=2^(app.n)
            app.B=app.B+1;
            app.UITable.Data=app.m;
        end




    else
        app.TabGroup.SelectedTab = app.HomeTab;
    end

end
```

```
% Button pushed function: Button_3
function Button_3Pushed(app, event)
  if app.inpDone





    %This button works same as the previous
    if app.B~=(2^app.n)+1
      app.m(app.B,end)=1;
    end
    if app.B<2^(app.n)
      app.OutputForEditField.Value = num2str(app.m(app.B+1,1:end-1));
    end
    if  app.B<=2^(app.n)
      app.B=app.B+1;
      app.UITable.Data=app.m;
    end
  else
    app.TabGroup.SelectedTab = app.HomeTab;
  end
end


% Button pushed function: GenerateOutputExpressionButton
function GenerateOutputExpressionButtonPushed(app, event)
  if app.inpDone
```

```
%Printing output expression for valid
%input by SUM OF PRODUCS rule
k=0;
str=' ';
for i=1:2^app.n

  if app.m(i,end)==1
    if k==1
      str =[str ' + '];
    end
  for j = 1:app.n
    if app.m(i,j)==0

      str = [ str '(~'];
      str = [ str char(64+j) ];
      str = [ str ')'];

    elseif app.m(i,j)==1

      str=[ str char(64+j)];
    end
    k=1;
  end
  end
end
```

```
    %special case when all outputs are zeros or ones
        if strcmp(str,' ')
            app.OutputExp.Value='0';
        elseif sum(app.m(:,end))==2^(app.n)
            app.OutputExp.Value='1';
        else
            app.OutputExp.Value= str;
        end




    else
        app.TabGroup.SelectedTab = app.HomeTab;
    end


end


% Button pushed function: ResetButton
function ResetButtonPushed(app, event)


    %This button initializes everything
    app.inpDone=0;
```

```
            app.B=1;

            app.m(:,end)=0;

            app.UITable.Data=app.m;

            app.OutputForEditField.Value =  num2str(zeros(1,app.n));

            app.TabGroup.SelectedTab = app.HomeTab;

            app.OutputExp.Value= " ";

            app.plotguide.Value= " ";

            hold(app.graph,'off')

            plot(app.graph,1,1);

            hold(app.graph,'off')

            app.UITable.Data=0;

            app.guide.Value= "First input the number of logical inputs and click the button";

        end



        % Button pushed function: GenerateLogicCircuitButton

        function GenerateLogicCircuitButtonPushed(app, event)

    if app.inpDone %This button will draw circuit for valid input









      %Special case when all outputs are zeros

      if sum(app.m(:,end))==0



        x=(0:1:8);
    y=ones(1,length(x));

    plot(app.graph,x,y,'m');

    xlim(app.graph,[-1, 9]);

    ylim(app.graph,[0, 1.2]);

    legend(app.graph,"0");

    app.plotguide.Value="The output is 0"; %Message to user









%Special case when all outputs are zeros

elseif sum(app.m(:,end))== 2^(app.N.Value)

    x=(0:1:8);

    y=ones(1,length(x));

    plot(app.graph,x,y,'m');

    xlim(app.graph,[-1, 9]);

    ylim(app.graph,[0, 1.2]);

    legend(app.graph,"1");

    app.plotguide.Value="The output is 1"; %Message to user









%Other than special cases

else
```

```
n=app.N.Value;

plotg= 'Inputs are ';



%Message to user

for i=1:n

   plotg=[plotg char(64+i) ' '];

end

plotg=[plotg 'serially.'];

app.plotguide.Value= plotg;








t=sum(app.m(:,end)); %Counting how many ones in output

xp=0:36;

yp=-2:0.1:2;

k=1;

a=[];







%This loop creates vectors to plot

%Connections between input lines and their complement lines
```

```
for i=1:2^n
    if app.m(i,end)==1
        for j = 1:n
            if app.m(i,j)==0
                start=-17.5+(j-1)*5;
            elseif app.m(i,j)==1
                start=-20+(j-1)*5;
            end
            a(k,:)= linspace(start,2*n+2,3);
            k=k+1;
        end
    end
end
```

```
%This loops draws input lines and there complements
for i=1:n
    y1=15:-1:-t*32;
    x1=i*ones(1,length(y1));
    plot(app.graph,5*x1-25,y1,'r');
    legend(app.graph,'off');
    axis([-30 80 -80 30])
    xlim(app.graph,[-30, 80]);
    ylim(app.graph,[-80, 30]);
    hold(app.graph,'on')
    y2=6.6:-1:-t*32 ;
    x2=i*ones(1,length(y2))+0.5;
    plot(app.graph,5*x2-25,y2,'m');
    y3=[10:-1:8] ;
    x3=i*ones(1,length(y3))+0.5;
    plot(app.graph,5*x3-25,y3,'r');
    x4=i+0.25:0.05:i+0.25+0.5;
    y4=4*abs(x4-i-0.5)+7;
    plot(app.graph,5*x4-25,y4,'b');
    r=0.2;
    th = 0:0.1:2*pi;
    xunit = r * cos(th);
    yunit = r * sin(th);
    plot(app.graph,xunit+((5*x4(1)-25)+(5*x4(end)-25))/2, yunit+y4(1)-1.2,'b');
    yh=y3(end)*ones(1,length(x4));
    plot(app.graph,5*x4-25,yh,'b');
    x5=i:0.1:i+0.5;
    y5=y3(1)*ones(1,length(x5));
    plot(app.graph,5*x5-25,y5,'r');
end


k=1;
```

```
%this loop draws AND gates and their connection lines

for i=0:t-1

    y6=(0:-1:-8);

    x6=(2*n)*ones(1,length(y6))+2;

    plot(app.graph,x6,y6-i*32,'b');

    plot(app.graph,a(k,:),zeros(length(a(k,:)))+y6(1)-i*32,'m')

    k=k+1;

    plot(app.graph,a(k,:),zeros(length(a(k,:)))+y6(end)-i*32,'m')

    k=k+1;

    yc=-4:0.1:4;

    xc=sqrt(16-(yc).^2)+x6(1);

    plot(app.graph,xc,yc+(y6(1)+y6(end))/2 - i*32,'b')

    plot(app.graph,xc-4,zeros(1,length(xc))- i*32,'m')

    plot(app.graph,xc-4,zeros(1,length(xc))- i*32-8,'m')



        %Draws additional AND gates if needed

        for j=1:n-2

            plot(app.graph,x6+j*8,y6-i*32-j*4,'b');

            plot(app.graph,a(k,:),zeros(length(a(k,:)))+y6(end)-i*32-j*4,'m')

            k=k+1;

            plot(app.graph,xc+j*8,yc+(y6(1)+y6(end))/2 - i*32-j*4,'b')

            plot(app.graph,xc+j*8,yc+(y6(1)+y6(end))/2 - i*32-j*4,'b')

            plot(app.graph,xc-4+j*8,zeros(1,length(xc))- i*32-j*4,'m')

            plot(app.graph,linspace(0,x6(1)+j*8,5),zeros(1,5)- i*32-8-j*4,'m')

        end

end
```

```
        %This loop draws OR gates and their connection lines

        for i=0:t-2

          for k=1:t-1

            xp1=-yp.^2;

            xp2=-0.1*yp.^2-3.6;

            plot(app.graph,xp1+(n-2)*10+10+(i+1)*40,2*yp-4*i-8-(n-2)*4,'b')

            plot(app.graph,xp2+(n-2)*10+10+(i+1)*40,2*yp-4*i-8-(n-2)*4,'b')

            plot(app.graph,yp+(n-2)*10+44+40*(k-1),zeros(1,length(yp))- k*4-8 -(n-2)*4,'m')

            plot(app.graph,xp+(n-2)*10+10+40*(k-1),zeros(1,length(xp))- k*4-0-(n-2)*4,'m')

            s=xp(1)+(n-2)*10+10;

            f=yp(1)+(n-2)*10+44+40*(k-1);

            xpp=s:f;

            plot(app.graph,xpp,zeros(1,length(xpp))- (k-1)*(32)-36-(n-2)*4,'m')

            yh = (-(k-1)*(32)-36-(n-2)*4):(- k*4-8-(n-2)*4);

            plot(app.graph,zeros(1,length(yh))+xpp(end),yh,'m')

          end

        end


        plot(app.graph,xp+(n-2)*10+10+40*(t-1),zeros(1,length(xp))- t*4-0-(n-2)*4,'m')

        close(1);

    end

else

     app.TabGroup.SelectedTab = app.HomeTab; %if input is not valid

end

        end
```

```
 % Button pushed function: UpdateTableandViewOutputExpressionButton
 function UpdateTableandViewOutputExpressionButtonPushed(app, event)
    app.TabGroup.SelectedTab = app.OutputExpressionTab;
 end



 % Button pushed function: NextButton
 function NextButtonPushed(app, event)
     app.TabGroup.SelectedTab =app.LogicCircuitTab;
 end



 % Button pushed function: BacktoHomeButton
 function BacktoHomeButtonPushed(app, event)
    app.TabGroup.SelectedTab =app.HomeTab;
 end



 % Button pushed function: BackButton
 function BackButtonPushed(app, event)
    app.TabGroup.SelectedTab =app.ViewTable;
 end
end
```

Discussion:

The MATLAB code is only shown for edited public properties and callback functions. The program can find output expressions from the truth table, but it does not simplify the output expression. So, using this program can sometimes cause a complex output for a simple circuit. This is the major drawback of this project. Again, there might be some bugs or unintended mistakes and an apology is expected.