1-1-1986

Instructions for Using Logic

John W. Scott

Dartmouth College

INSTRUCTIONS FOR USING *LOGIC*

John W. Scott

Technical Report PCS-TR86-102

<u>Instructions for using LOGIC</u>

John W. Scott
Department of Mathematics and Computer Science
Dartmouth College
Hanover, NH 03755

Logic is a digital logic simulator. It's application, however, is not specific to digital logic. It provides a useful way of investigating boolean logic in any discipline. The user can create logic diagrams and test their results. Such diagrams provide a graphic display of logical expressions. Where logical expressions can become unwieldy quickly as they grow, diagrams are easy to comprehend at a much greater levels of complexity. This application's ability to easily display the results of all of the subexpressions can be particularly helpful in understanding the expression.

Instead of character boolean operators digital logic uses gates. These gates are icon representations of the same boolean operators. Thus, there is an icon representing AND, one representing OR, etc.. Each gate has one output (result) that can be either true or false, depending on the value of its input(s). Wires are used to link the subexpressions to their operators. The results of expressions are called sinks and the operand variables are called sources. Thus in "A = B or C" A is a sink and B and C are sources.

These diagrams are created with a MacPaint-like interface. The user selects gates from the palette and enters them into a document. These gates may then be connected by choosing the diagonal line in the palette. Connections are always drawn from the output to the input. They are deleted by drawing over an existing connection while in the same mode. Any input that does not have a connection has a default value of false. Gates are deleted, along with all of their connections, by selecting the hammer from the palette. This operation, like most of the others, is not undo-able so caution is advised. Selecting the wire cutters from the palette allows the user to delete all of a gate's connections simultaneously, without destroying the gate.

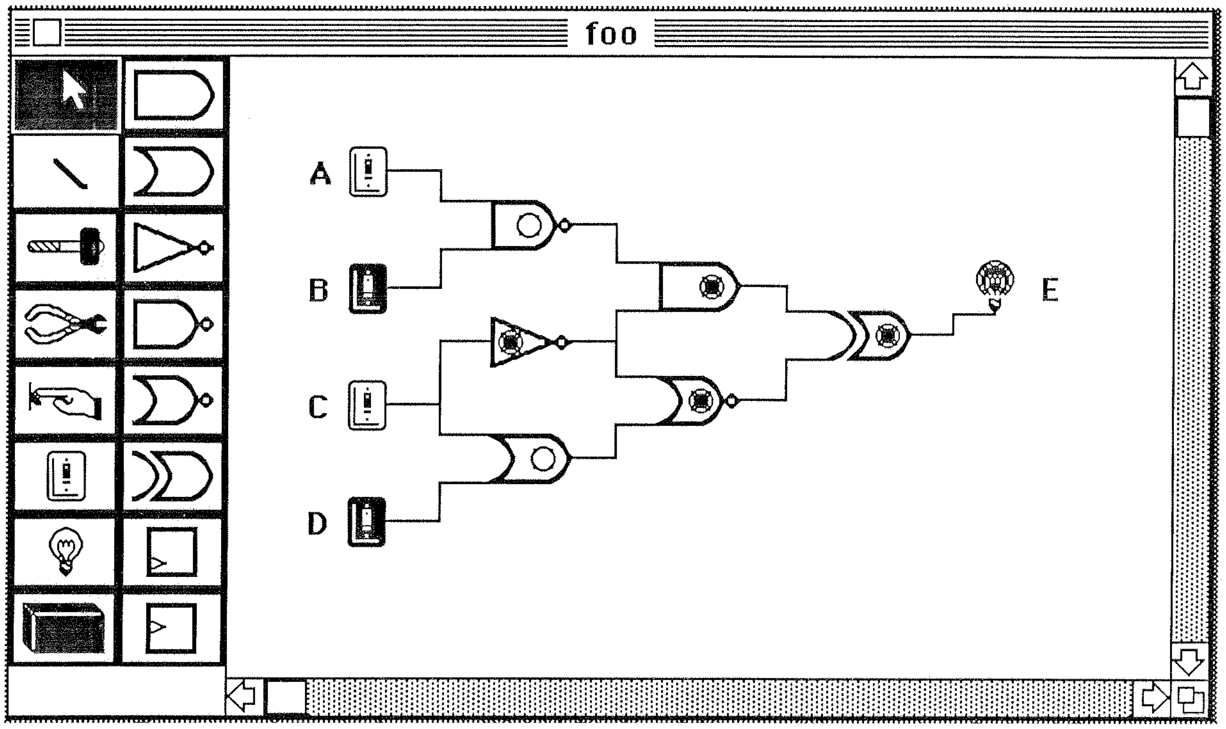
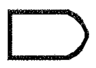


figure 1

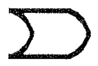
Sources for the circuits are provided in the form of light switch icons. These have optional character labels used in identifying the source in an equation. These sources may be turned on and off by selecting the finger icon from the palette and clicking on the source. Sinks are represented by light bulbs and also have optional labels. For the purposes of this application white = on = true = 1 and black = off = false = 0. In the document above sources A and C are true while B, D and the sink E are all false.

There are six types of basic logic gates provided. These are illustrated in figure 2 along with their truth tables. These gates may be connected in any fashion as long as no loop is created and each gate's output is connected to no more than ten inputs (maximum fanout of 10).



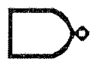
And Gate

	0	1
0	0	0
1	0	1



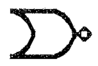
Or Gate

	0	1
0	0	1
1	1	1



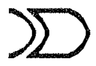
Nand Gate

	0	1
0	1	1
1	1	0



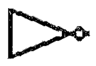
Nor Gate

	0	1
0	1	0
1	0	0



Xor Gate

	0	1
0	0	1
1	1	0



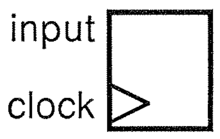
Inverter

0	1
1	0

figure 2

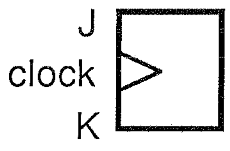
There are also two generic types of flip-flops provided. The first is a positive edged d-type flip-flop. Upon an positive edge (0 -> 1 transition) at the clock (the lower input) the value at the top input is stored. The output remains stable at all other times. A basic J-K flip-flop is also provided. The J input is located in the top left corner, the K input in the lower left with the clock in between. This gate is negative edged (1 -> 0). Upon a negative transition from the clock the output toggles (inverts) if both J and K are true. If J and K are false the output holds, If only J is true the output is set (true) and if only K is true the output is cleared (false). Note that conventionally an inverted output is also supplied for these flip-flops. This can be simulated with an inverter.

D-Type flip-flop



clock	0	1
↓	H	H
↑	0	1

J-K flip-flop



clock	**~J ~K**	**J K**	**J~K**	**~J K**
↓	H	T	1	0
↑	H	H	H	H

H = Hold previous output T = Toggle previous output

figure 3

The final type of gate provided is a four input black box. These black boxes simulate programmable logic arrays (PLA). The PLAs are "programmed" upon their creation by duplicating the truth table in the truth table view. There is more information on the programming in the comments on the truth table view.

Options:

Show Gate Values: the value of the output is shown for all gates. The value is indicated in a circle inside the gate. Figure 1 shows an example of this option.

Show Equations: shows the logical expressions associated with circuits. When the value of a cicuit's sink (result) changes, the logical expression that determines that sink is shown in a window in the table view. These expressions are only useful for relatively small circuits. The nesting becomes complex with more than five or six gates. These expressions are truncated if they are longer than 60 - 80 characters. Figure 4 shows the expression for the gates in figure 1.

Show Labels: displays the single character names for the sources and sinks. These names are used in the logical expressions generated by the "Show Equations" option.

Right Angles: draws perpendicular connecting lines rather than the normal rubber band lines. This can be confusing if the circuits are complex or if they are not layed out neatly. Figure 1 demonstrates this option.

Show Selections: indicates which gates are currently selected by drawing rectangles around their extent. This is normally used in conjunction with clean up operations and the evaluate command where gate selection is important.

Evaluate: creates a truth table for selected gates. The table is then displayed along with its karnaugh map in the truth table view. A table may be created for a black box simply by selecting that black box and chosing evaluate from the menu. Alternatively a complete circuit may be evaluated. In this second case the user selects the gates in the cicuit (possibly by chosing "Select All" from the menu) and then choosing evaluate from the menu. There must be exactly one sink selected for the result and fewer than five sources selected. It is not appropriate to include a flip-flop in a truth table evaluation as they depend on transitions as well as discrete values. There is a one to one correspondence between the sink and the table output and between the sources and the table inputs. If fewer than four sources are selected, any extra high order inputs (W, X...) are left as "don't cares" (they do not affect the output). The evaluate command provides a convenient way of cloning selected gates into a black box. Figure 4 shows the evaluation of all of the gates in figure 1.

Information: provides help on the selected palette item. To help save resources some items are grouped together on help dialogs.

Clean-Up:

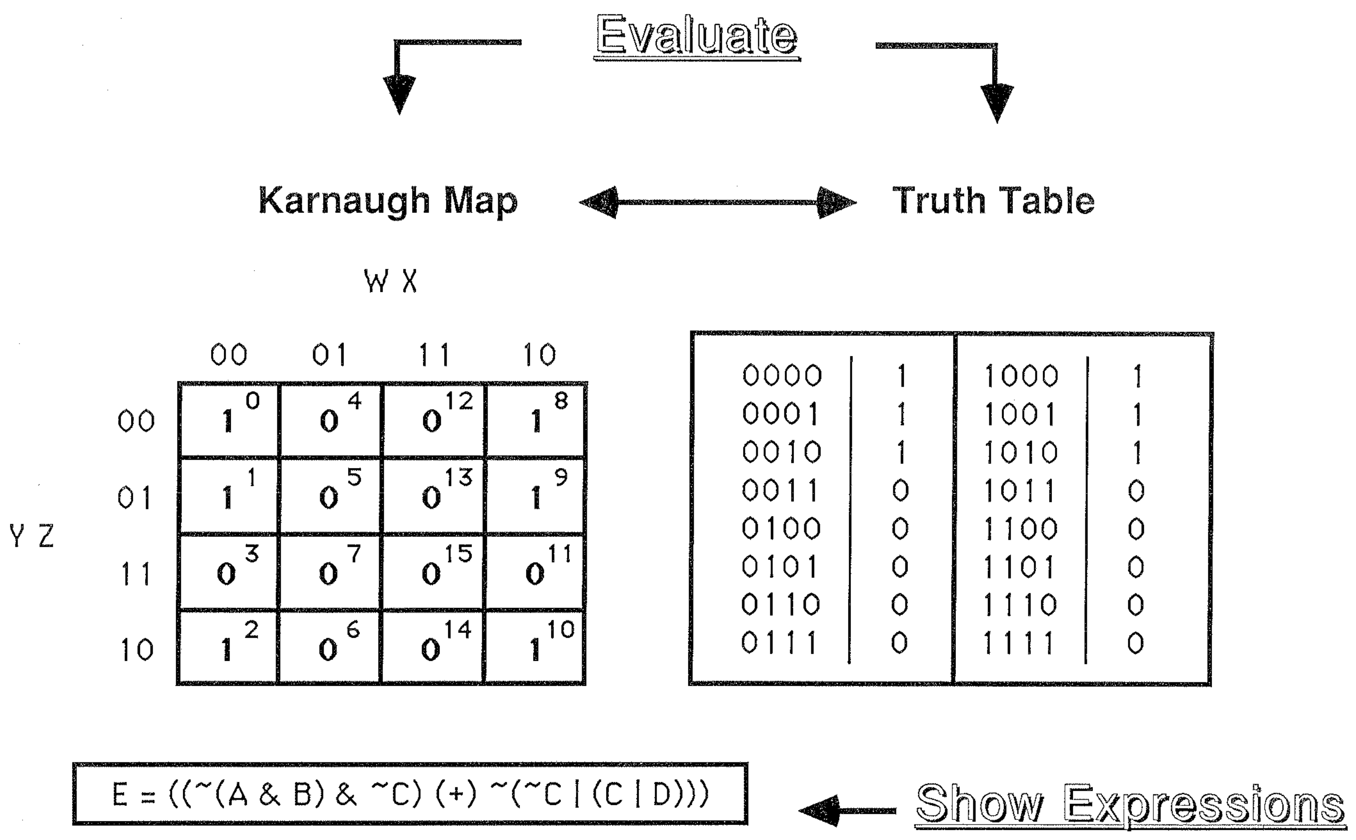
This menu provides basic clean-up operations for the document. Align Vertical aligns all selected gates to their furthest left extent. Align Horizontal aligns all selected gates to their highest extent. The spread options evens the spacing between selected gates, either horizontally or vertically. Finally, in case or error, there is an undo option.

Truth Table View:

The truth table view is used for three purposes: 1) to evaluate selected gates in the document view, 2) to show logical expressions, and 3) to program black boxes. The first two have already been discussed above in the "Options" section.

The programming of black boxes (PLAs) may be done very simply by evaluating selected gates and then creating a black box. In this manner the black box will duplicate the logic of the selected gates. Black boxes may be more direcly customized by editing the truth table. This is done by clicking the mouse on the value that you wish to change. This may be done either in the karnaugh map or in the truth table. Clicking in one will also change the other.

The truth table and karnaugh map are two ways of looking at the same information. The truth table clearly shows the output values for each combination of input values. The least significant bit corresponds to the top source. The karnaugh map shows the outputs in a manner convenient for reducing the table's logical expression. Between any two adjacent cells only one of the four input values changes. Thus, if two adjacent (not diagonal) cells are both true then the single changing input becomes a "don't care" and may be dropped from the sub-expression. Note that the concept of adjacency wraps around. Thus the top row is adjacent to the bottom and the left is adjacent to the right. In the karnaugh map the top source corresponds to Z, the next lowest source to Y etc.. Consult a book on digital logic for more information on the use of karnaugh maps.



Y Z \ W X	00	01	11	10
00	1^{0}	0^{4}	0^{12}	1^{8}
01	1^{1}	0^{5}	0^{13}	1^{9}
11	0^{3}	0^{7}	0^{15}	0^{11}
10	1^{2}	0^{6}	0^{14}	1^{10}

0000	1	1000	1
0001	1	1001	1
0010	1	1010	1
0011	0	1011	0
0100	0	1100	0
0101	0	1101	0
0110	0	1110	0
0111	0	1111	0

E = ((~(A & B) & ~C) (+) ~(~C | (C | D)))

figure 4