Lab 1 - 135 mins / 3 lectures.

Main objects

A. Intro to Logisim
B. How to build a circuit for Boolean function?
C. Analys a circuit.
D. How to build a circuit from a given Truth Table?
E. 7-segments LED and make the IC to convert BCD in to 7-segments LED.

Resources

Object	Software / File	Book	Slide	Student's task
A. Software	https://sourceforge.net/projects/circuit/files/latest/download			Download and install to your PCs
Hello circuit	Lab1_1.circ	[2] Beginner's tutorial	Ch.3 p.6	Experiment
Half adder	Lab1_2.circ	[1] p.367	Ch.3 p.6	Experiment
Input / Output	Lab1_3.circ	[2] Library Reference		Experiment
7-segments LED	Lab1_4.circ (just part e)	[1] p.122		Finish all

Preferences

[1] Brian Holdsworth, Clive Woods, [2002], Digital Logic Design, 4th edition, Newnes, Oxford.

[2] http://www.cburch.com/logisim/docs/2.5.0/en/guide/index.html access on Jan 12th.

Some useful links

[3] http://www.uize.com/examples/seven-segment-display.html

[4] https://www.dcode.fr/7-segment-display

Table of Contents

Table of figures.

Part 1. Welcome to Logisim!

This part is copied from Logisim's "Beginner's tutorial".

Logisim allows you to design and simulate digital circuits. It is intended as an educational tool, to help you learn how circuits work.

To practice using Logisim, let's build a XOR circuit - that is, a circuit that takes two inputs (which we'll call *x* and *y*) and outputs 0 if the inputs are the same and 1 if they are different. The following truth table illustrates.

x	*y*	*x* XOR *y*
0	0	0
0	1	1
1	0	1
1	1	0

Figure 1. Truth table of XOR gate.

We might design such a circuit on paper.

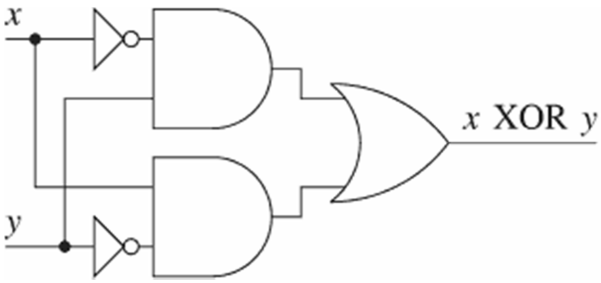


Figure 2 . Circuit of XOR gate.

But just because it's on paper doesn't mean it's right. To verify our work, we'll draw it in Logisim and test it. As an added bonus, we'll get a circuit that's looks nicer than what you probably would draw by hand.

Step 0: Orienting yourself

When you start Logisim, you'll see a window similar to the following. Since you'll be using a different system, some of the details may be slightly different.

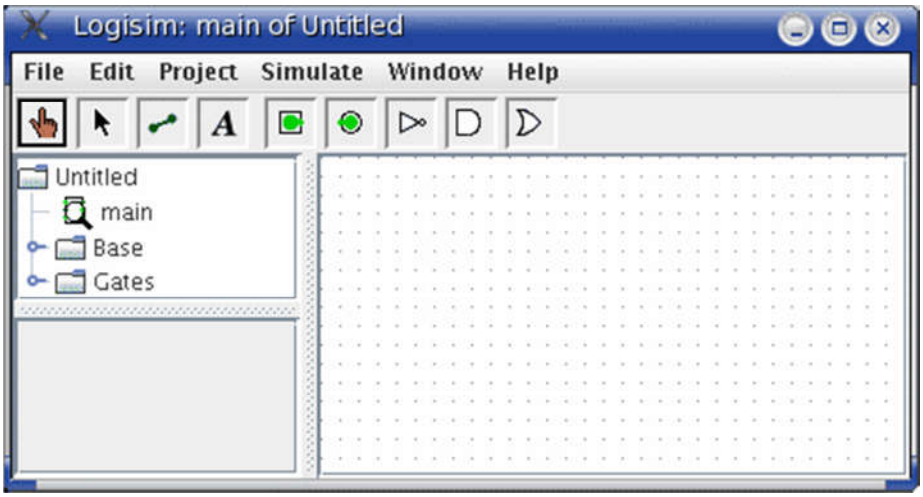


Figure 3. Main window of Logisim.

All Logisim is divided into three parts, called the *explorer pane*, the *attribute table*, and the *canvas*. Above these parts are the *menu bar* and the *toolbar*.

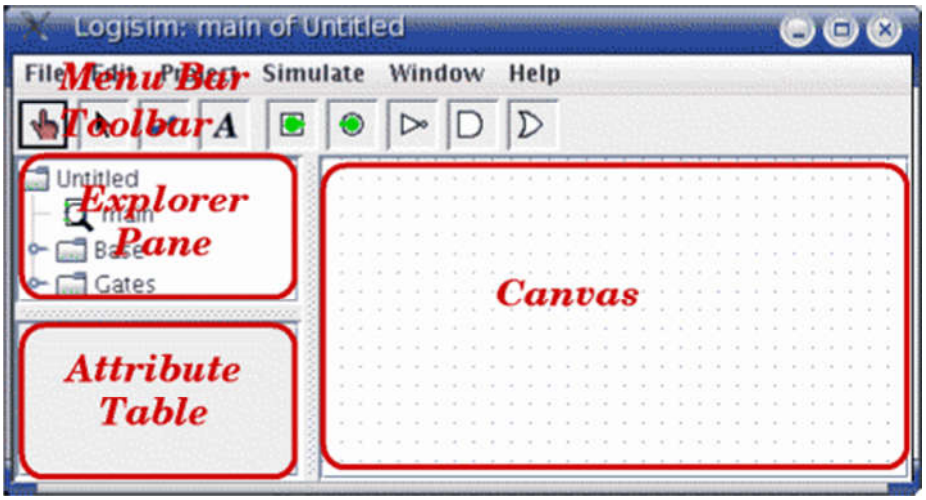


Figure 4. Important zones of your workplace.

We can quickly dispose of the explorer pane and the attribute table: We won't be examining them in this tutorial, and you can just ignore them. Also, the menu bar is self-explanatory.

That leaves the toolbar and the canvas. The canvas is where you'll draw your circuit; and the toolbar contains the tools that you'll use to accomplish this.

Step 1: Adding gates

Recall that we're trying to build the following circuit in Logisim.

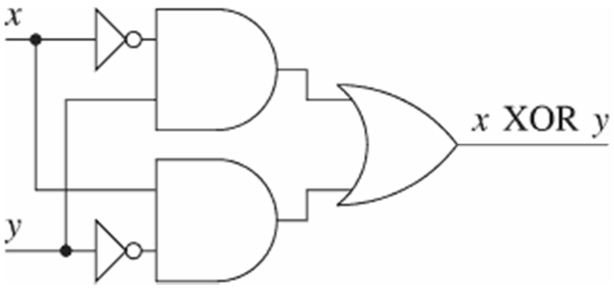


Figure 5. Circuite of XOR gate.

Building a circuit is easiest by inserting the gates first as a sort of skeleton for connecting wires into the circuit later. The first thing we're going to do is to add the two AND gates. Click on the AND tool in the toolbar (, the next-to-last tool listed). Then click in the editing area where you want the AND gates to go. Be sure to leave plenty of room for stuff on the left.

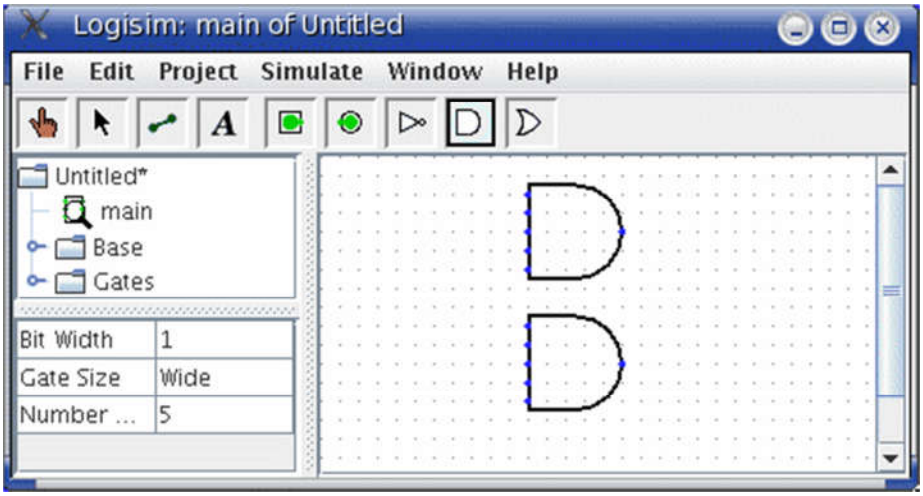


Figure 6. Select two AND gates and put them in your Canvas

Notice the five dots on the left side of the AND gate. These are spots where wires can be attached. It happens that we'll just use two of them for our XOR circuit; but for other circuits, you may find that having more than two wires going to an AND gate is useful.

Now add the other gates. First click on the OR tool ( ); then click where you want it. And select the NOT tool ( ) and put those two gates into the canvas.

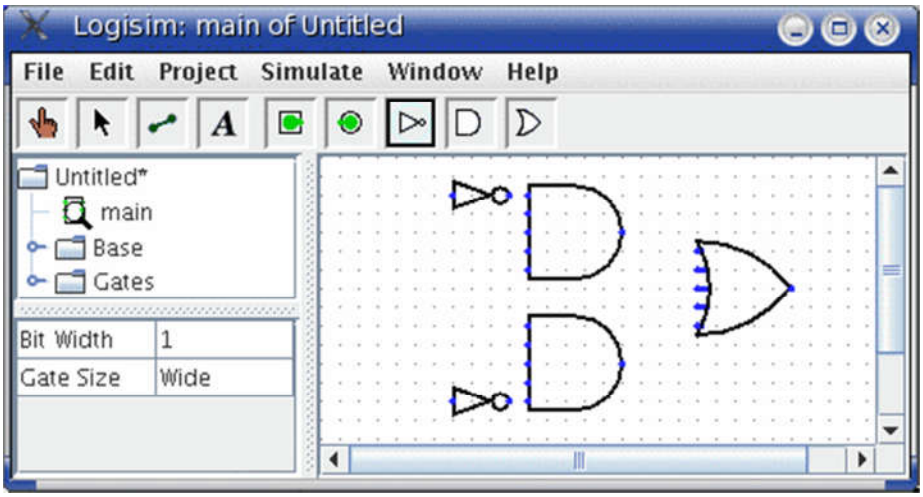


Figure 7. Add more gates: NOT and OR.

I left a little space between the NOT gates and the AND gates; if you want to, though, you can put them up against each other and save yourself the effort of drawing a wire in later.

Now we want to add the two inputs *x* and *y* into the diagram. Select the input pin (), and place the pins down. You should also place an output pin () next to the OR gate's output. (Again, though I'm leaving a bit of space between the OR gate and the output pin, you might choose to place them right next to each other.)

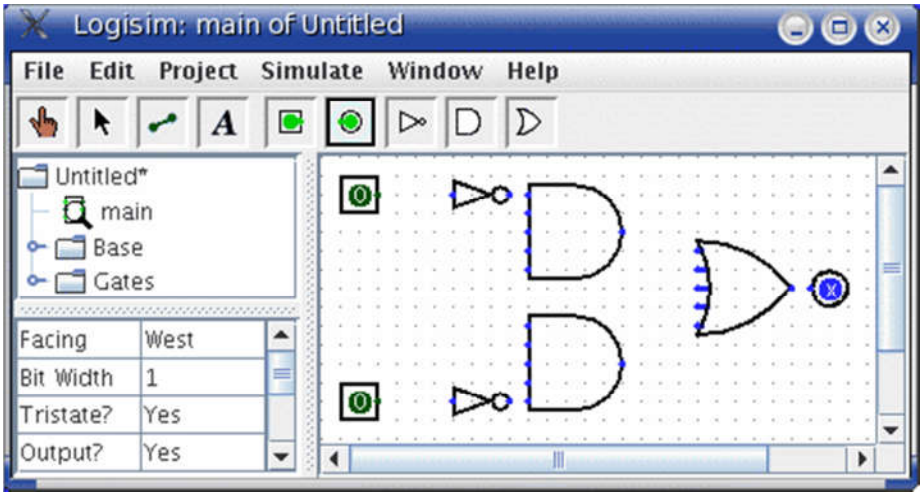


Figure 8. Add two input pins and one output pin.

If you decide you don't like where you placed something, then you can right-click (or control-click) anything in the canvas to view a pop-up menu. Choose Delete. You can also rearrange things using the select tool ().

Step 2: Adding wires

After you have all the components blocked out on the canvas, you're ready to start adding wires. Select the wiring tool (). Then start dragging from one position to another in the canvas area, and a wire will start to appear between the two points.

The wiring tool is pretty intelligent. Whenever a wire ends at another wire, Logisim automatically connects them. You can also "extend" or "shorten" a wire by dragging one of its endpoints using the wiring tool.

Wires in Logisim must be horizontal or vertical. To connect the upper input to the NOT gate and the AND gate, then, I added three different wires.

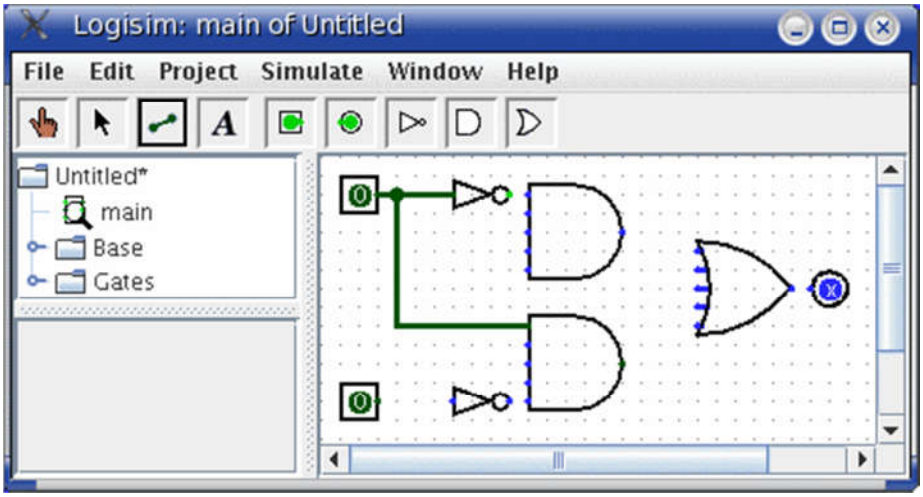


Figure 9. Wiring your gates together.

Logisim automatically connects wires to the gates and to each other. This includes automatically drawing the circle at a *T* intersection as above, indicating that the wires are connected.

As you draw wires, you may see some blue or gray wires. Blue in Logisim indicates that the value at that point is ``unknown", and gray indicates that the wire is not connected to anything. This is not a big deal temporarily. But by the time you finish your circuit, none of your wires should be blue or gray. (The unconnected legs of the OR gate will still be blue: That's fine.)

If you do have a blue or a gray wire after you think everything ought to be connected, then something is going wrong. It's important that you connect wires to the right places. Logisim draws little dots on the components to indicate where wires ought to connect. As you proceed, you'll see the dots turn from blue to light or dark green.

Once you have all the wires connected, all of the wires you inserted will themselves be light or dark green.

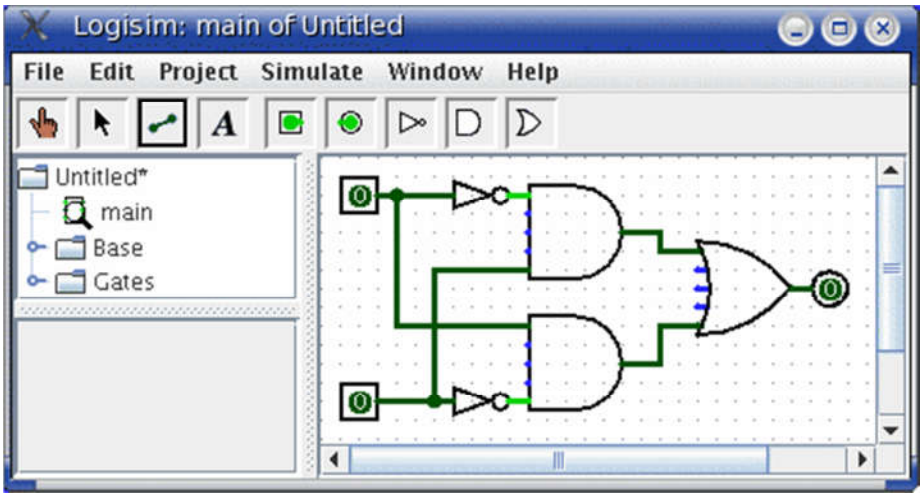
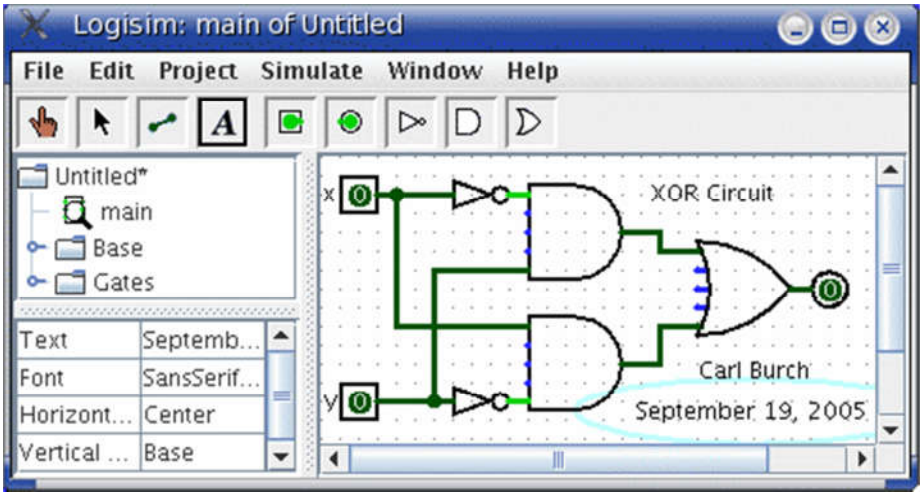


Figure 10. Completed XOR circuit.

Step 3: Adding text (Labeling)

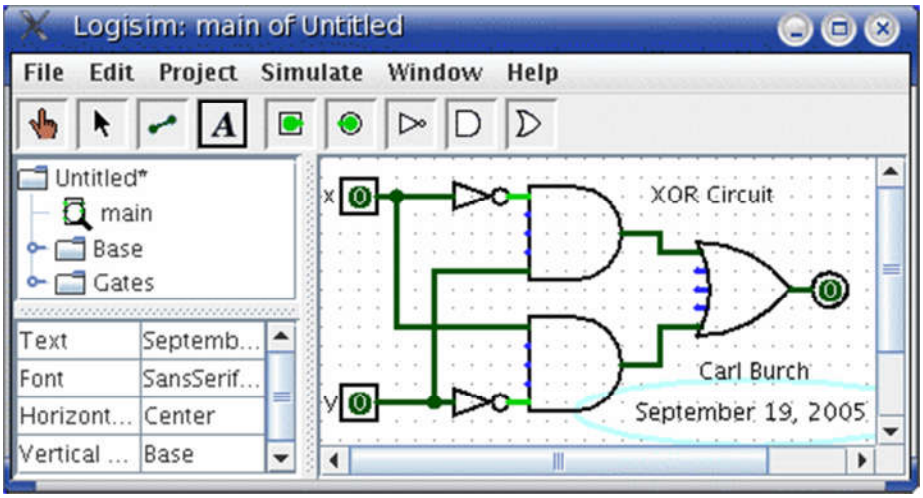
Adding text to the circuit isn't necessary to make it work; but if you want to show your circuit to somebody (like a teacher), then some labels help to to communicate the purpose of the different pieces of your circuit.

Select the text tool (**A**). You can click on an input pin and start typing to give it a label. (It's better to click directly on the input pin than to click where you want the text to go, because then the label will move with the pin.) You can do the same for the output pin. Or you could just click any old place and start typing to put a label anywhere else.



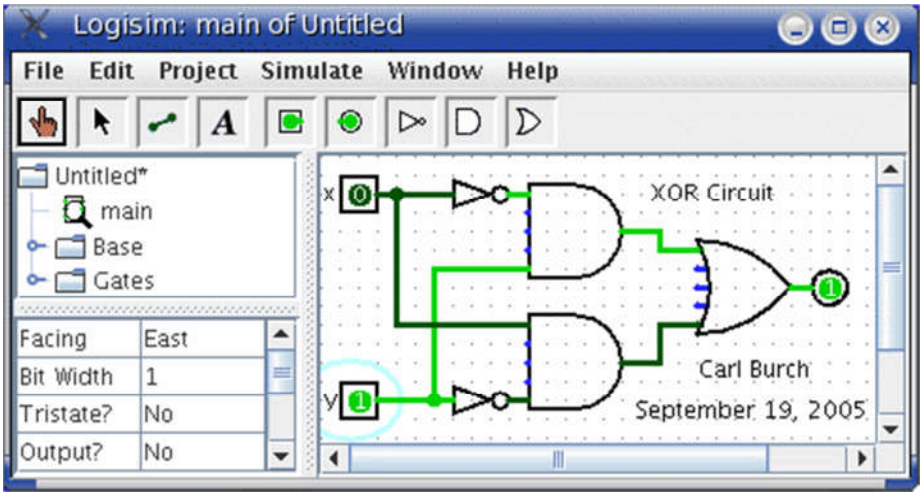
Step 4: Testing your circuit

Our final step is to test our circuit to ensure that it really does what we intended. Logisim is already simulating the circuit. Let's look again at where we were.



Note that the input pins both contain 0s; and so does the output pin. This already tells us that the circuit already computes a 0 when both inputs are 0.

Now to try another combination of inputs. Select the poke tool () and start poking the inputs by clicking on them. Each time you poke an input, its value will toggle. For example, we might first poke the bottom input.



When you change the input value, Logisim will show you what values travel down the wires by drawing them light green to indicate a 1 value or dark green (almost black) to indicate a 0 value. You can also see that the output value has changed to 1.

So far, we have tested the first two rows of our truth table, and the outputs (0 and 1) match the desired outputs.

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

By poking the switches through different combinations, we can verify the other two rows. If they all match, then we're done: The circuit works!

To archive your completed work, you might want to save or print your circuit. The File menu allows this, and of course it also allows you to exit Logisim. But why quit now?

Now that you are finished with tutorial, you can experiment with Logisim by building your own circuits. If you want to build circuits with more sophisticated features, then you should navigate through the rest of the help system to see what else you can do. Logisim is a powerful program, allowing you to build up and test huge circuits; this step-by-step process just scratches the surface.

Part 2. Your first circuit

Half adder

- How many inputs?
- How many output?
- What is the function of each output?
- Each output needs a circuit!

Z = A XOR B

Cout = A AND B

Fill the blank using above functions.

A	B	Cout	Z
0	0		
0	1		
1	0		
1	1		
0	0		
0	1		
1	0		
1	1		

Build it yourself.

Be carefully when:

- Pick a gate from Explorer panel.
- Put in PINs (input and output are diffence!) and label them.
- Wiring them together.
- Test your final circuit, but how to do it?

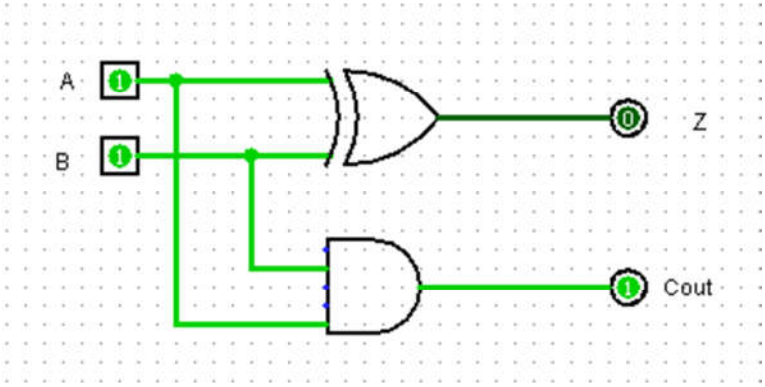


Figure 11. Circuit of a Half Adder.

Ex 1: Do your second circuit with below given function, build the Truth Table by calculate with Boolean Theorems and another Table from experiments and compare them.

a. $X = A.(B + \bar{C})$

b. $Y = (A \quad B) \quad C$

c. $Z = \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C}$

Why do we call "Full Adder"? Coz the adding includes the "carry in" bit, which result from the adding lower bits, value while a "Half Adder" doesn't.

Clearly,

0+0=0, carry out = 0.

1+1=0, but carry out = 1.

Part 3. How to input?

a. Button

Pin: using Poke to toggle its value between 0 and 1. Pin can be set to n bits.

Button: using Poke to press it. The output value is 1 when button is pressed, and becomes 0 when button is released.

b. Joystick

Use Poke tool to move it. There's two outputs which give out the value of coordinates of X and Y axis. This value could be chosen from 2 to 5 bits.

			100 001			
			100 010			
			100 011			
001 100	010 100	011 100	100 100	101 100	110 100	111 100
			100 101			

			100 110			
			100 111			111 111

Figure 12. Value of outputs X and Y with a 3-bits Joystick.

? Ex 2: Experiment with Button and Joystick, try Joystick with 2-bits and 5-bits output and write down what is the diffent?

c. Keyboard

Behavior: This component allows the circuit to read keys typed from the keyboard - as long as the keys are representable in the 7-bit ASCII code. After clicking the component using the poke tool, the user can type characters, which accumulate in a buffer. At all times, the ASCII value for the leftmost character in the buffer is sent out the rightmost output. When the clock input is triggered, the leftmost character disappears from the buffer and the new leftmost character is sent on the rightmost output.

The supported characters for the buffer include all the printable ASCII characters, as well as space, newline, backspace, and control-L. In addition, the left-arrow and right-arrow keys move the cursor within the buffer, and the delete key deletes the character to the right of the cursor (if any).

The component is asynchronous in the sense that when the buffer is empty and the user types a character, that character is sent immediately as an output, without any wait for a clock pulse.

Pins:

- **Clock** - when triggered while the read-enable pin isn't 0, the leftmost character from the buffer is deleted, and the outputs are updated to reflect the buffer's new status.
- **Read Enable** - when 1 (or floating or error), a clock edge will consume the leftmost character from the buffer. The clock input is ignored when Read Enable is 0.
- **Clear -** when 1, the buffer is emptied and does not accept further characters.
- **Available** - this is 1 when the buffer contains at least one character and 0 when the buffer is empty.
- **Data** - the 7-bit ASCII code for the leftmost character in the buffer, or 0 if the buffer is empty.

Attributes

- Buffer Length: The number of characters that the buffer can hold at once.
- Trigger: If the value is Rising Edge, then when the clock input changes from 0 to 1, the leftmost character is consumed (when enabled by the Read Enable input). If it is Falling Edge,, then this happens when the clock input changes from 1 to 0.

? **Ex 3: Open your Lab1_3.circ, there's a keyboard with all pins, clock, buttons connected. Using Poke tool to select that keyboard, typing with your real keyboard some characters one by one. After each character, which value is shown at output pin? Read more about "ASCII" code.**

Part 4. How to output?

This part is copied from Logisim's manual

a. Pin out

b. LED

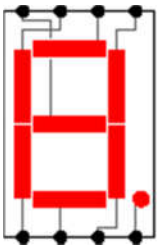
c. Built-in Hex Digit

d. Led 7 segment.

7-segments LED

Behavior

Displays the values of its eight one-bit inputs. Segments are either colored or light gray depending on the inputs. The correspondence is as follows.



(Manufacturers vary as to how they map inputs to segments; the correspondence used here is based on Texas Instruments' TIL321.)

Pins

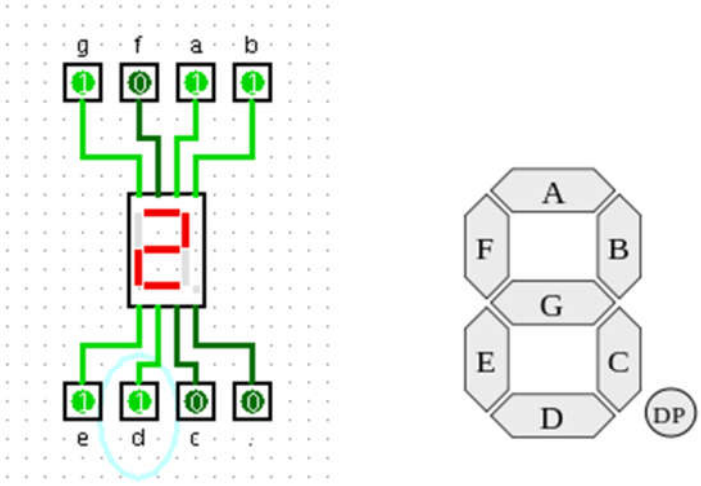
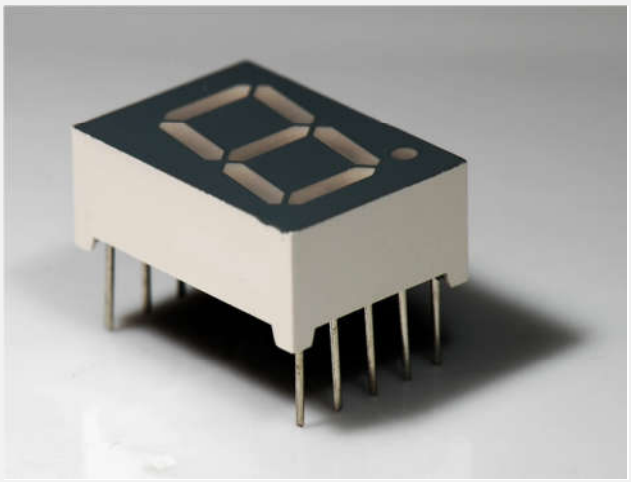


Figure 13. Pins layout of a 7-segments LED.

In fact, a 7-segments LED has 10 pins. The 2 extra pins are common negative (or positive) to make the led on or off or blink.



Attributes

- On Color: The color with which to draw the display segments and decimal point when they are on.
- Off Color: The color with which to draw the display segments and decimal point when they are off.
- Background: The color with which to draw the display's background (transparent by default).
- Active On High? If yes, then the segments light when the corresponding input is 1. If no, they light when the corresponding input is 0.

e. Led Matrix

Behavior

Displays a small grid of pixels, whose values are determined by the current inputs. The grid can have up to 32 rows and 32 columns.

Pins

The interface to the component varies depending on the value of the Input Format attribute. It has three possible values.

- **Columns**: The inputs are lined along the component's south edge, with one multibit input for each column of the matrix. Each input has as many bits as there are rows in the matrix, with the low-order bit corresponding to the southmost pixel in the column. A 1 indicates to light the corresponding

pixel, while a 0 indicates to keep the pixel dim. If any of the bits for a column are either floating or error values, then all pixels in the column are lit.

- **Rows**: The inputs are lined along the component's west edge, with one multibit input for each row of the matrix. Each input has as many bits as there are columns in the matrix, with the low-order bit corresponding to the rightmost pixel in the row. As with the Columns format, a 1 indicates to light the corresponding pixel, and a 0 indicates to keep the pixel dim. If any bits for a row are floating or error values, then all pixels in the row are lit.
- Select Rows/Columns: There are two inputs on the component's west edge. The upper multibit input has as many bits as there are columns in the matrix, with the low-order bit corresponding to the rightmost column. The lower multibit input has as many bits as there are rows in the matrix, with the low-order bit corresponding to the bottom row. If any bits in either input are floating or error values, all pixels in the matrix are lit. Normally, though, a pixel at a particular row-column location is lit if the corresponding column bit in the upper input is 1 and the corresponding row bit in the lower input is 1. For example, for a 5x7 matrix, if the first input is 01010 and the second is 0111010, then the second and fourth columns are lit for the second, third, fourth, and sixth rows; the result appears to be a pair of exclamation points. (This input format may seem unintuitive, but LED matrixes are sold commercially with exactly this interface. Lite-On sells such components, for example.)

Attributes

Input Format (read-only after component is created)

Selects how the pins correspond to pixels, as outlined above.

Matrix Columns

Selects how many columns are in the matrix, which may range from 1 up to 32.

Matrix Rows

Selects how many rows are in the matrix, which may range from 1 up to 32.

On Color

Selects the color of a pixel when it is lit.

Off Color

Selects the color of a pixel when it is dim.

Light Persistence

When this is other than 0, a pixel that is lit remains lit for the given number of clock ticks after the component's inputs indicate that the pixel should become dim.

Dot Shape

The square option means that each pixel is drawn as a 10x10 square, filling the component with no gaps between pixels. The circle option means that each pixel is drawn as a diameter-8 circle, with gaps between each circle. The circle option is more difficult to interpret, but it more closely approximates the off-the-shelf LED matrix components.

? **Ex 4: Using Led Matrix 6 x 8 to draw the first lettet in your Last name**

Part 5: Subcircuits.

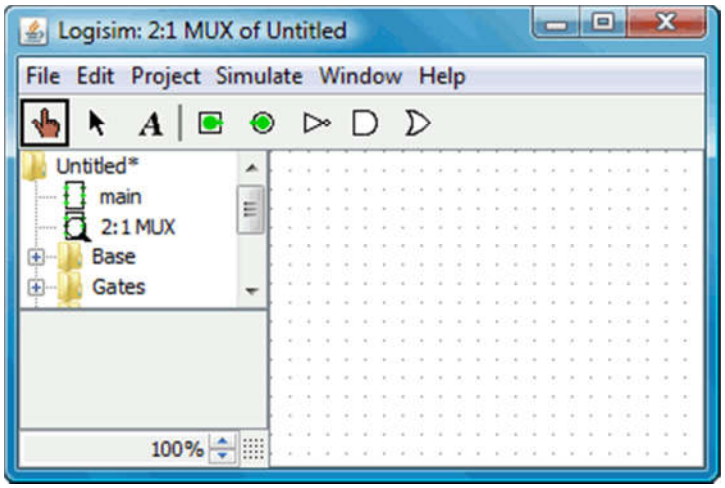
Subcircuits

As you build circuits that are more and more sophisticated, you will want to build smaller circuits that you can use multiple times as a module nested within larger circuits. In Logisim, such a smaller circuit that is used in a larger circuit is called a **subcircuit**.

If you're familiar with computer programming, you're familiar with the subprogram concept, whether it's called a *subroutine*, *function*, *method*, or *procedure* in your favored language. The subcircuit concept is analogous to this, and it serves the same purpose: To break a large job into bite-sized pieces, to save the effort of defining the same concept multiple times, and to facilitate debugging.

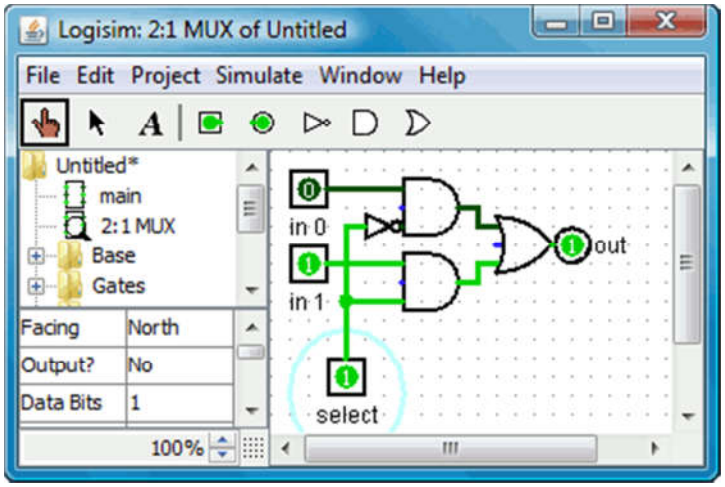
Every Logisim project is actually a library of circuits. In its simplest form, each project has only one circuit (called "main" by default), but it is easy to add more: Select Add Circuit... from the Project menu, and type any name you like for the new circuit you want to create.

Suppose we want to build a 2-to-1 multiplexer named "2:1 MUX." After adding the circuit, Logisim will look like this.



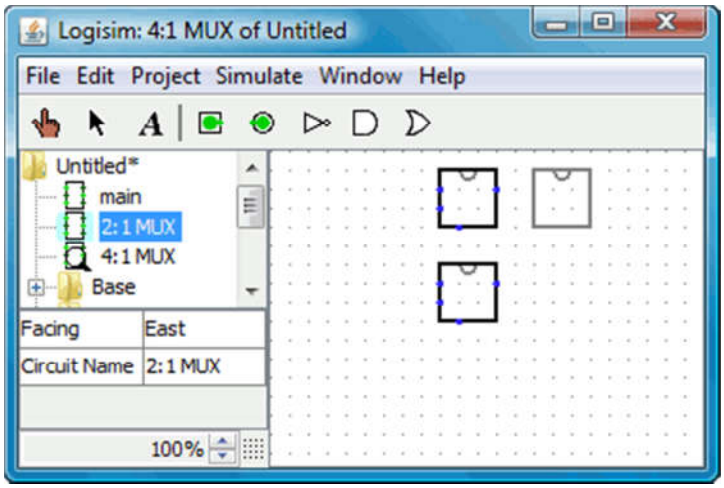
In the explorer pane, you can now see that the project now contains two circuits, "main", and "2:1 MUX." Logisim draws a magnifying glass over the icon of the circuit currently being viewed; the current circuit name also appears in the window's title bar.

After editing the circuit to appear like a 2:1 multiplexer, we might end up with the following circuit.



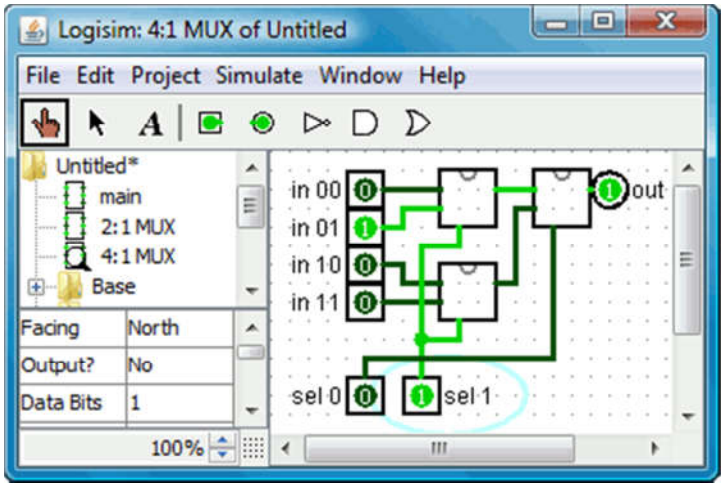
Using subcircuits

Now suppose we want to build a 4-to-1 multiplexer using instances of our 2-to-1 multiplexer. Of course, we would first create a new circuit, which we'll call "4:1 MUX." To add 2-to-1 multiplexers into our circuit, we click the 2:1 MUX circuit *once* in the explorer pane to select it as a tool, and then we can add copies of it, represented as boxes, by clicking within the canvas.



If you were to double-click the 2:1 MUX circuit in the explorer pane, then the window would switch to editing the 2:1 MUX circuit instead.

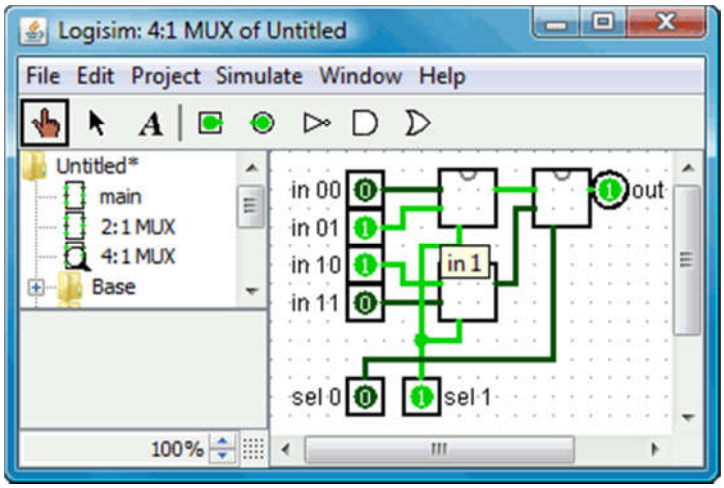
After building up the circuit, we end up with the following.



Our circuit for a 4-to-1 multiplexer uses three copies of the 2-to-1 multiplexer, each drawn as a box with pins along the side. The pins on this box correspond to the input and output pins in the 2:1 MUX circuit. The two pins on the west side of the box correspond to the two pins that face east in the 2:1 MUX circuit; the pin on the box's east side corresponds to the 2:1 MUX's west-facing pin (which happens to be an output pin); and the pin on the box's south side corresponds to the 2:1 MUX's north-facing pin. The order of the two pins on the box's west side correspond to the same top-down ordering from the subcircuit's design. (If there were several pins on the box's north or south side, they would correspond to the same left-right order in the subcircuit.)

If the pins in the subcircuit's layout have labels associated with them, then Logisim will display that label in a **tip** (that is, a temporary text box) when the user

hovers the mouse over the corresponding location of the subcircuit component. (If you find these tips irritating, you can disable them via the Project Options window's Canvas tab.)



Several other components will display these tips, too: For some of the pins of a built-in flip-flop, for example, hovering over it explains what that pin does.

Incidentally, every pin to a circuit must be either an input or an output. Many manufactured chips have pins that behave as an input in some situations and as an output in others; you cannot construct such chips within Logisim (at least, in the current version).

Logisim will maintain different state information for all subcircuits appearing in a circuit. For example, if a circuit contains a flip-flop, and that circuit is used as a subcircuit several times, then each subcircuit's flip-flop will have its own value when simulating the larger circuit.

Now that we have the 4-to-1 multiplexer defined, we can now use it in other circuits. Logisim has no limits on how deeply circuits can be nested - though it will object to nesting circuits within themselves!

Note: There's nothing wrong with editing a circuit that is being used as a subcircuit; in fact, this is quite common. Be aware, though, that any changes to a circuit's pins (adding, deleting, or moving them) will rearrange them also in the containing circuit. Thus, if you change any pins in a circuit, you will also need to edit any circuits using it as a subcircuit.

Default appearance

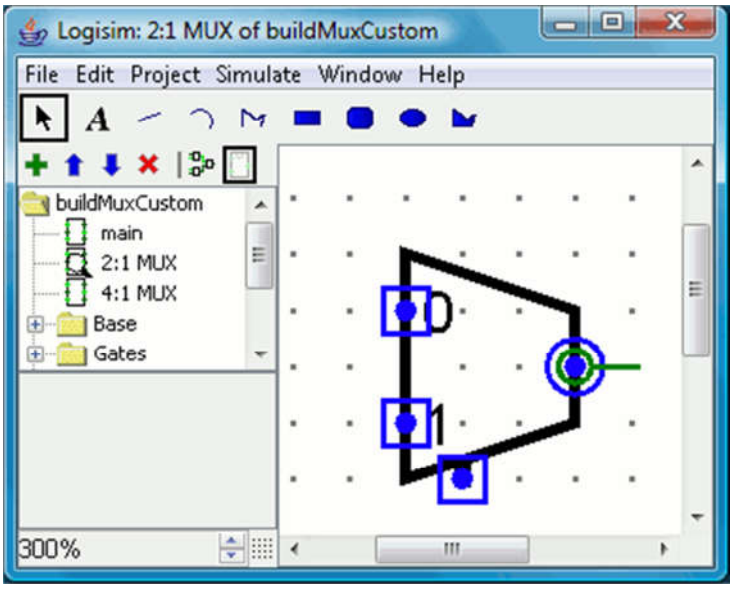
By default, when a subcircuit is placed within a larger circuit, it is drawn as a rectangle with a notch indicating the north end of the subcircuit's layout. Pins will be placed on the rectangle's border based on their facing: Pins that face east in the

layout (and typically appear on the west side of the layout) will be placed on the rectangle's west side, according to their top-down ordering in the layout. Pins that face south in the layout (typically toward the north side of the layout) will be placed on the rectangle's north side, according to the left-to-right ordering in the layout.

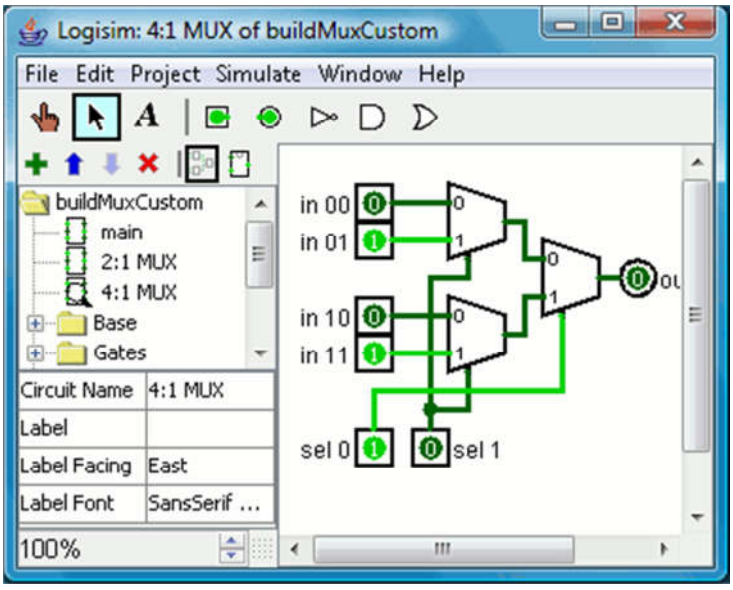
The default rectangle can optionally include some letters that will appear in the middle of the rectangle. To specify this, select the selection tool ( ) and click the background of the circuit's layout. This will show the circuit attributes in the attribute table, including the Label, Label Facing, and Label Font attributes. The value of the Label attribute will be drawn in the rectangle's center; the Label Facing attribute customizes which direction the text is drawn, and of course the Label Font attribute customizes the font used.

Customized appearance

The default appearance is very usable, and indeed Logisim existed for many years with no other option. If, however, you prefer that the subcircuit be drawn differently, you can select Edit Circuit Appearance from the Project menu, and Logisim's interface will switch from its regular layout-editing interface to an interface for drawing the circuit's appearance. Below, we are editing the 2:1 multiplexer's appearance so that it is drawn with the usual trapezoid rather than a rectangle. (You can see the project toolbar just below the regular toolbar. This can be enabled through the Project menu, and it allows quick switching between editing the layout and appearance.)



With the appearance for the 2:1 multiplexer drawn as above, the layout for the 4:1 multiplexer would then appear as the following.



The appearance editor is like a traditional drawing program, but there are a few special symbols for indicating how the drawing works when placed into a circuit's layout. These special symbols cannot be removed.

- The green circle with a line coming out of it, which we'll call the anchor. There is exactly one anchor in each subcircuit appearance. Each component in a circuit has a single point identifying its location; a user sees this when creating a new component: The mouse click identifies just a single location, and the component is placed relative to that (usually with the primary output at the mouse's location) The anchor identifies the mouse's location relative to the overall drawing when the subcircuit is created.

  The anchor also identifies the appearance's facing, as indicated by the direction the anchor's line points from its circle. When placing the subcircuit into a layout, the user can change the subcircuit's facing; the anchor's facing indicates in which direction the appearance is oriented. In our example, the anchor is facing east, and each instance of the subcircuit in the 4:1 multiplexer is also facing east, so they are all drawn in the same orientation as the 2:1 multiplexer's appearance.

- The blue circles and squares with dots in them are the subcircuit's ports. There are exactly as many ports as there are input and output pins in the circuit. Ports corresponding to inputs are drawn as squares, while ports corresponding to outputs are drawn as circles. Each port indicates how a wire connecting into the circuit will correspond to an input or output pin within the layout.

  When you select a port, Logisim will indicate the corresponding pin by popping up a miniature diagram of the layout in the window's bottom

right corner, with the corresponding pin(s) drawn in blue. This does not happen when all ports are selected.

The toolbar contains tools for adding additional shapes, as listed below with descriptions of how the shift and alt key modifies the tool behavior. In addition, clicking or dragging the mouse with the control key pressed regularly snaps the mouse position to the nearest grid point.

Select, move, copy, and paste shapes.

Add or edit text.

Create a line segment. Shift-drag keeps the line's angle at a multiple of 45°.

Create a quadratic Bezier curve. For the first drag, where you specify the curve's endpoints, shift-drag keeps the endpoints at an angle that is a multiple of 45°. Then you click to indicate the control point's location; shift-click ensures the curve is symmetric, while alt-click draws the curve through the control point.

Create a sequence of connected lines, whose vertices are indicated by a succession of clicks. Shift-clicking ensures that the angle between the previous vertex and the current one is a multiple of 45°. Double-click or press the Enter key to complete the shape.

Create a rectangle through dragging from one corner to the opposite corner. Shift-drag to create a square, and alt-drag to create the rectangle starting from the center.

Create a rectangle with rounded corners through dragging from one corner to the opposite corner. Shift-drag to create a square, and alt-drag to create the rectangle starting from the center.

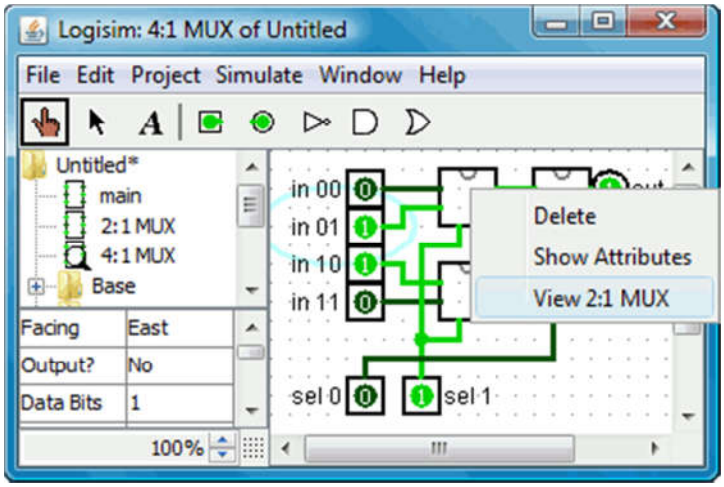
Create an oval through dragging from one corner of its bounding box to the opposite corner. Shift-drag to create a circle, and alt-drag to create the oval starting from the center.

Create an arbitrary polygon, whose vertices are indicated by a succession of clicks. Shift-clicking ensures that the vertex is at a 45° angle from the previous one. Double-click, press the Enter key, or click the starting vertex to complete the shape.

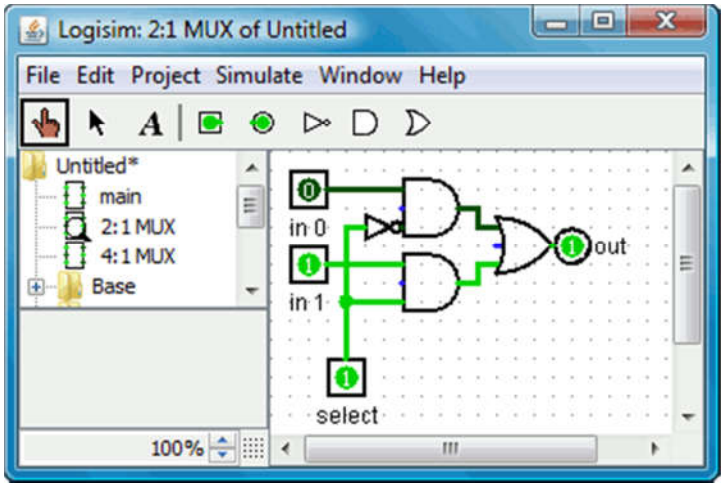
Debugging subcircuits

As you test larger circuits, you will likely find bugs. To nail down what's going wrong, exploring what's going on in the subcircuits while running the overall circuit can help. You can view the state of the subcircuit in two ways: First, you

can bring up the subcircuit's popup menu by right-clicking or control-clicking it, choosing the View option.



Or if you have the Poke tool selected, you can click the circuit, and a magnifying glass will appear over its center; double-clicking where the magnifying glass appears will also enter the subcircuit's state.

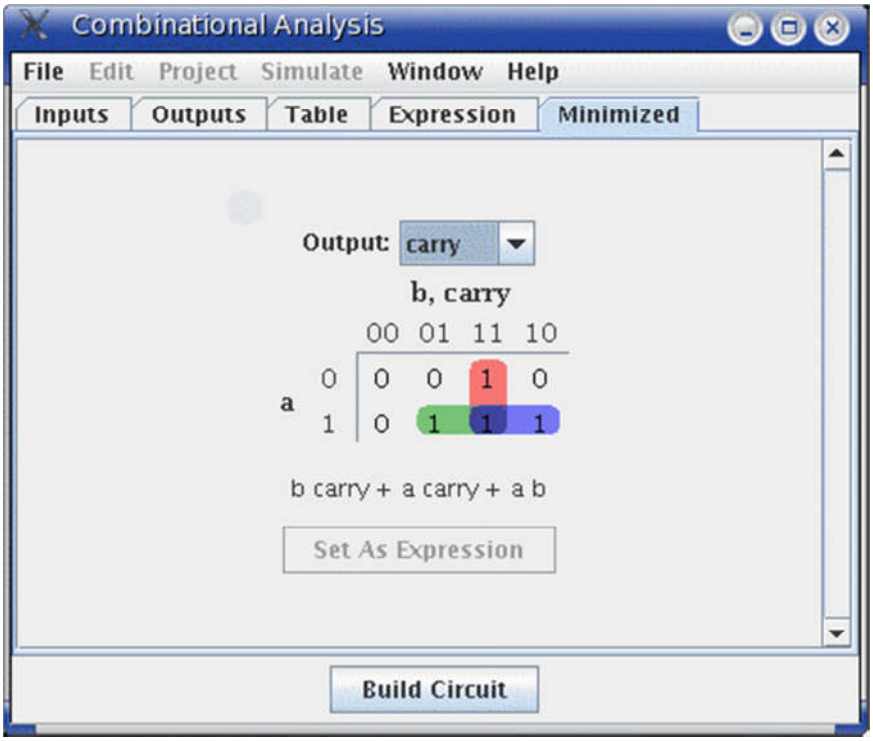


Notice that the pins' values in the subcircuit match the values being sent to them in its containing circuit.

While in the subcircuit, you are allowed to alter the circuit. If the changes affect any of the subcircuit's outputs, they are propagated into the containing circuit. One exception: The subcircuit inputs are determined based on the values coming into the circuit from the supercircuit, so it doesn't make sense to toggle those values. If you attempt to poke a subcircuit's input, a dialog will pop up asking, The pin is tied to the supercircuit state. Create a new circuit state? Clicking No will cancel the toggle request, while clicking Yes will create a copy of the viewed state, divorced from the outer circuit, with the input pin toggled.

Once you have completed viewing and/or editing, you can return to the parent circuit either by double-clicking the parent circuit in the explorer pane, or via the Go Out To State submenu of the Simulate menu.

Combinational analysis



All circuits fall into one of two well-known categories: In a **combinational circuit**, all circuit outputs are a strict *combination* of the current circuit inputs, whereas in a **sequential circuit**, some outputs may depend on past inputs (the *sequence* of inputs over time).

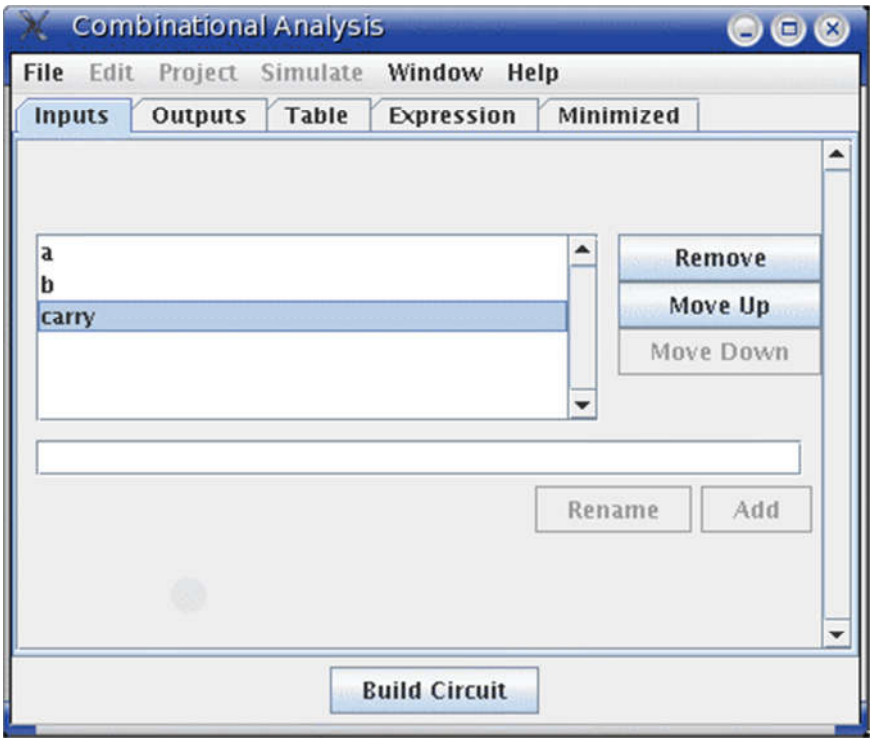
The category of combinational circuits is the simpler of the two. Practitioners use three major techniques for summarizing the behavior of such circuits.

- logic circuits
- Boolean expressions, which allow an algebraic representation of how the circuit works
- truth tables, which list all possible input combinations and the corresponding outputs

The *Combinational Analysis* module of Logisim allows you to convert between these three representations in all directions. It is a particularly handy way of creating and understanding circuits with a handful of one-bit inputs and outputs.

Editing the truth table

On opening the Combinational Analysis window, you will see that it consists of five tabs.



This page describes the first three tabs, Inputs, Outputs, and Table. The next page of the guide describes the last two tabs, Expression and Minimized.

The Inputs and Outputs tabs

The Inputs tab allows you to view and edit the list of inputs. To add new inputs, type it in the field at the pane's bottom, and click Add. If you want to rename an existing input, select it in the list in the pane's upper left region; then type the name and click Rename.

To remove an input, select it from the list and click Remove. You can also reorder the inputs (which affects the order of columns in the truth table and in the generated circuit) using the Move Up or Move Down buttons on an input.

All actions affect the truth table immediately.

The Outputs tab works in exactly the same way as the Inputs tab, except of course it works with the list of outputs instead.

The Table tab

The only item under the Table tab is the current truth table, diagrammed in the conventional order, with inputs constituting the columns on the left and outputs constituting the columns on the right.

You can edit the current values appearing in the output columns by clicking on the value of interest. The values will cycle through 0, 1, and *x* (representing a "don't care"). As we'll see on the next page, any don't-care values allow the computation of minimized expressions some flexibility.

You can also navigate and edit the truth table using the keyboard. And you can copy and paste values using the clipboard. The clipboard can be transferred to any application supporting tab-delimited text (such as a spreadsheet).

If the truth table is based on an existing circuit, you may see some pink squares in the output columns with "!!" in them. These correspond to errors that occurred while calculating the value for that row - either the circuit seemed to be oscillating, or the output value was an error value (which would be pictured as a red wire in the Logisim circuit). Hovering your mouse over the entry should bring up a tool tip describing which type of error it was. Once you click on the error entry, you will be in the 0-1-*x* cycle; there is no way to go back.

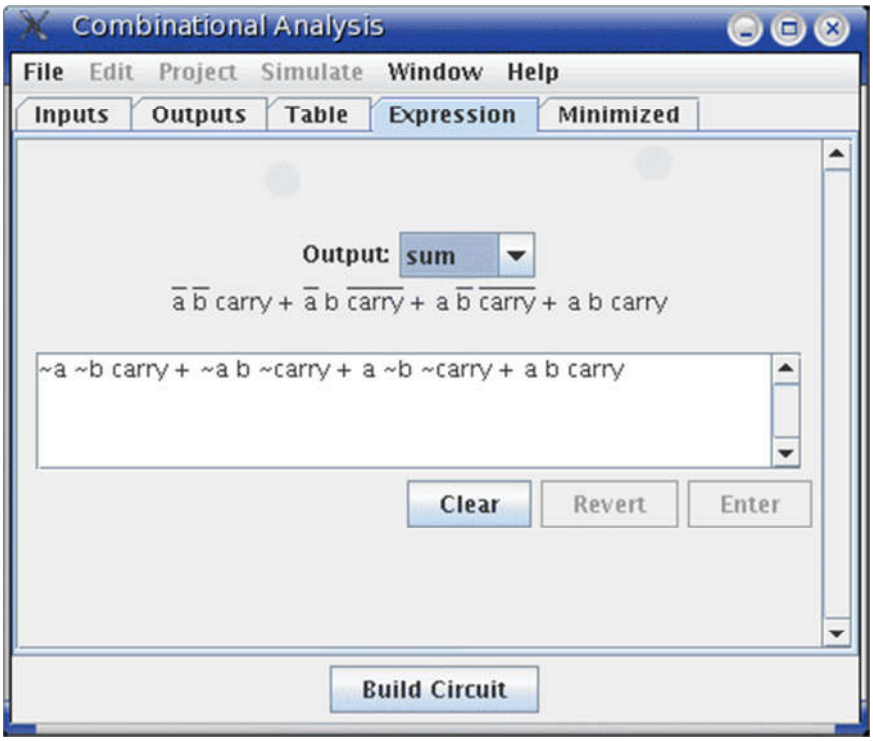
Creating expressions

For each output variable, the Combinational Analysis window maintains two structures - the relevant column of the truth table, and a Boolean expression - specifying how each output relates to its input. You can edit either the truth table or the expression; the other will automatically change as necessary to keep them consistent.

As we will see on the next page, the Boolean expressions are particularly useful because the Combinational Analysis window will use these when told to build a circuit corresponding to the current state.

You can view and edit the expressions using the window's last two tabs, the Expression tab and the Minimized tab.

The Expression tab



The Expression tab allows you to view and edit the current expression associated with each output variable. You can select the output expression you want to view and edit using the selector labeled "Output:" at the pane's top.

Just below the selector will appear the expression formatted in a particularly common notation, where an OR is represented as addition, an AND is represented as multiplication, and a NOT is denoted with a bar above the portion affected by the NOT.

The text pane below this displays the same information in ASCII form. Here, a NOT is represented with a tilde ('~').

You can edit the expression in the text pane and click the Enter button to make it take effect; doing this will also update the truth table to make it correspond. The Clear button clears the text pane, and the Revert button changes the pane back to representing the current expression.

Note that your edited expression will be lost if you edit the truth table.

In addition to multiplication and addition standing for AND and OR, an expression you type may contain any of C/Java logical operators, as well as simply the words themselves.

highest precedence	~ !	**NOT**
	(none) & &&	**AND**
	^	**XOR**
lowest precedence	+ | ||	**OR**

The following examples are all valid representations of the same expression. You could also mix the operators.

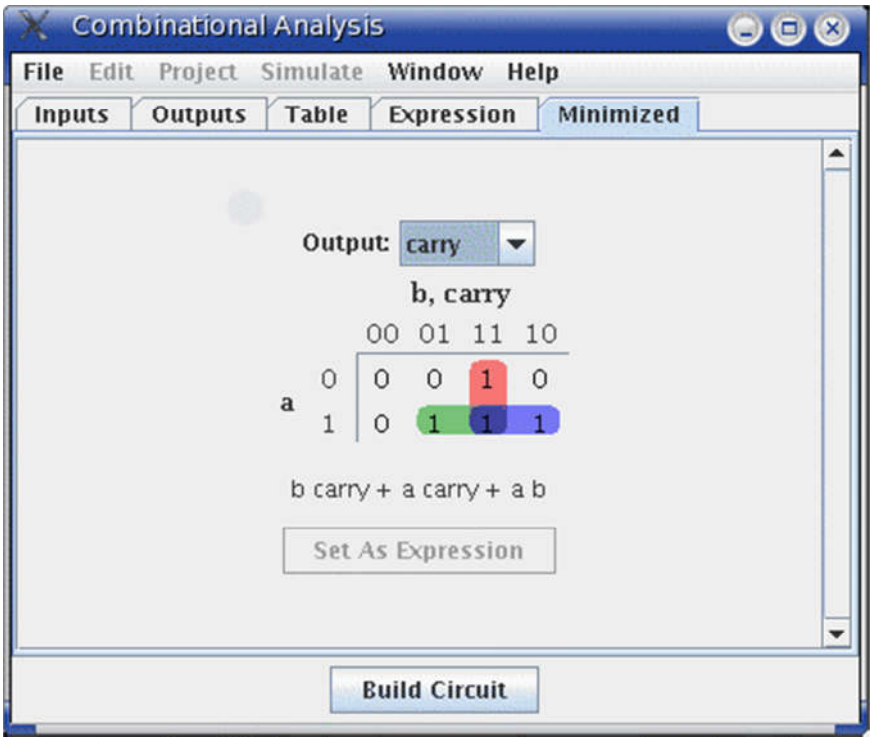
```
~a (b + c)

!a && (b || c)

NOT a AND (b OR c)
```

In general, parentheses within a sequence of ANDs (or ORs or XORs) do not matter. (In particular, when Logisim creates a corresponding circuit, it will ignore such parentheses.)

The Minimized tab



The final tab displays a minimized sum-of-products expression corresponding to a column of the truth table. You can select which output's minimized expression you want to view using the selector at top.

If there are four or fewer inputs, a Karnaugh map corresponding to the variable will appear below the selector. You can click the Karnaugh map to change the corresponding truth table values. The Karnaugh map will also display the currently selected terms for the minimized expression as solid semitransparent rounded rectangles.

Below this is the minimized expression itself, formatted as in the Expression tab's display. If there are more than four inputs, the Karnaugh map will not appear; but the minimized expression will still be computed. (Logisim uses the Quine-McCluskey algorithm to compute the minimized expression. This is equivalent to a Karnaugh map, but it applies to any number of input variables.)

The **Set As Expression** button allows you to select the minimized expression as the expression corresponding to the variable. This will generally not be necessary, as edits to the truth table result in using the minimized expression for the changed column; but if you enter an expression through the Expression tab, this can be a convenient way to switch to the corresponding minimized expression.

Generating a circuit

The Build Circuit button will construct a circuit whose gates correspond to the currently chosen expressions for each output. The circuit's inputs and outputs will be displayed in top-down order corresponding to how they appear under the Inputs and Outputs tabs. Generally speaking, the constructed circuit will be attractive; and, indeed, one application of Logisim's Combinational Analysis module is to beautify poorly drawn circuits. Still, as with any automatic formatting, it will not express the structural details that a human-drawn circuit would.

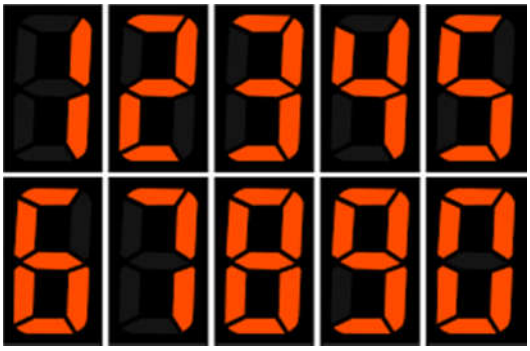
When you click the Build Circuit button, a dialog box will appear prompting you to choose which project where you want the circuit and the name you wish to give it. If you type the name of an existing circuit, then that circuit will be replaced (after Logisim prompts you to confirm that you really want to do this).

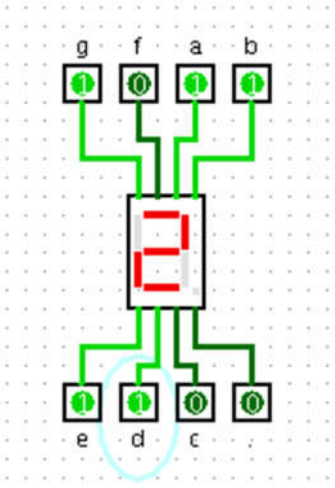
The Build Circuit dialog includes two options. The Use Two-Input Gates Only option specifies that you want all gates constructed to have two inputs. (NOT gates, of course, constitute an exception to this rule.) The Use NAND Gates Only option specifies that you would like it to translate the circuit into one using only NAND gates. You can select both options if you want to use only two-input NAND gates.

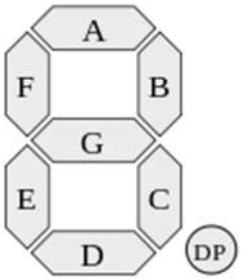
Logisim cannot construct a NAND-only circuit for an expression containing any XOR operators. This option will therefore be disabled if any outputs' expressions contain XORs.

Case study: How to built the Truth table of BCD to 7-seg LED IC?

- How many inputs do we get?
- How about outputs?
- How the Truth table look like?
- With each value of BCD, which value should the outputs give out?







X3	X2	X1	X0	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	0

Part 6. Make the circuit better.

Buses and Splitters

- Bus is using to transfer more than one bit in the line.
- Splitter is used when we need just some bit of a bus.
- Fan in = width of bus.
- Fan out = number of buses of output.
- Be careful with order of fan out. For example ABCD and DCBA is two diffent numbers.

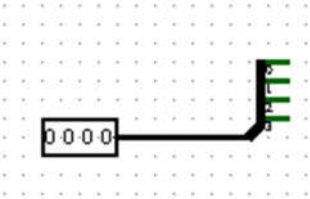


Figure 14. A pin 4 bits and a Spiltter 4 / 4.

Tunnels

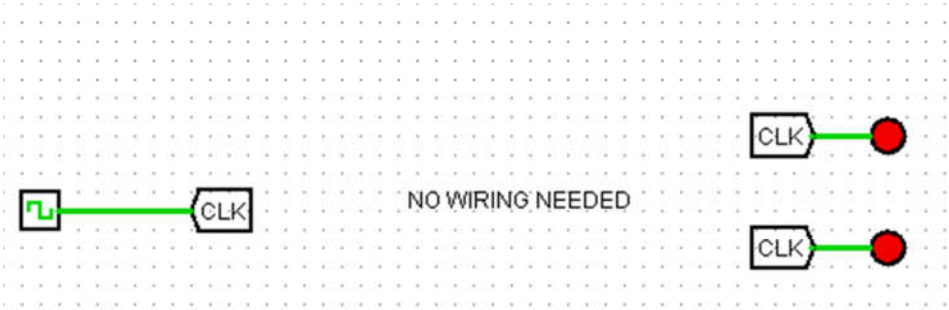


Figure 15. All tunnels which have the same label are connected!

Excercises

5. Translate some Boolean functions into circuit and test them.
6. We could build a circuit automatically by input its formula. Try some in the The Expression tab which was introduced in Part 4.
7. In 7-seg LED, there some “don’t care” led. For example, we can use 2 types of number 7 as below picture. Look for “don’t care” term and re-build the IC in Part 5.

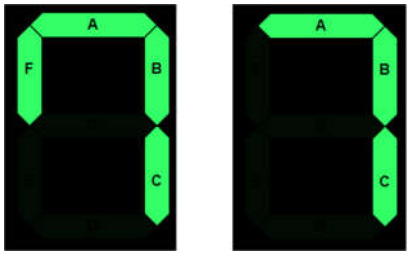


Figure 16. Two ways to show number 7.

8. Build a circuit with a joystick and matrix led 5x5. The matrix should show the position of the joystick in realtime.
9. Package all the circuits you’ve done in Ex 1, 3, 4 into separated sub-circuits and using them in a bigger circuit.