

·库参考手册

Logisim 库包含一组允许您通过在画布区域中单击并拖动鼠标来与电路交互的工具。通常，工具用于将特定类型的组件添加到电路中；但一些最重要的工具，例如戳工具 (Poke Tool) 和选择工具 (Select Tool)，允许您以其他方式与组件交互。

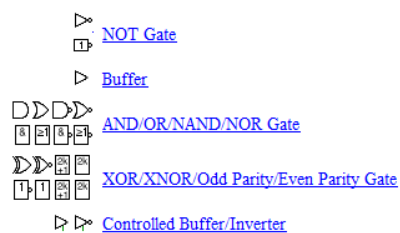
Logisim 内置库可分为 7 大类，其包含的所有工具/组件都以文件夹的形式记录在本参考材料中。

- 导线库：与导线直接交互的元件。
- 逻辑门库：执行简单逻辑功能的部件。
- 复用器库：更复杂的组合组件，如多路选择和解码器。
- 运算器库：执行算术的组件。
- 存储器库：记忆数据的组件，如触发器、寄存器和 RAM。
- 输入输出库：为与用户交互而存在的组件。
- 基础库：使用 Logisim 不可或缺的工具。

导线库



逻辑门库



复用器库










运算器库





存储器库

-  [Adder](#)
-  [Subtractor](#)
-  [Multiplier](#)
-  [Divider](#)
-  [Negator](#)
-  [Comparator](#)
-  [Shifter](#)
-  [Bit Adder](#)
-  [Bit Finder](#)

输入输出库

-  [D/T/J-K/S-R Flip-Flop](#)
-  [Register](#)
-  [Counter](#)
-  [Shift Register](#)
-  [Random](#)
-  [RAM](#)
-  [ROM](#)












基础库

-  [Button](#)
-  [Joystick](#)
-  [Keyboard](#)
-  [LED](#)
-  [7-Segment Display](#)
-  [Hex Digit Display](#)
-  [LED Matrix](#)
-  [TTY](#)


-  [Poke Tool](#)
-  [Edit Tool](#)
-  [Select Tool](#)
-  [Wiring Tool](#)
-  [Text Tool](#)
-  [Menu Tool](#)
-  [Label](#)

第一节：导线库


导线库主要包含与导线和基本电路概念相关的元件。

-  [Splitter](#)
-  [Pin](#)
-  [Probe](#)
-  [Tunnel](#)
-  [Pull Resistor](#)
-  [Clock](#)
-  [Constant](#)
-  [Power/Ground](#)
-  [Transistor](#)
-  [Transmission Gate](#)
-  [Bit Extender](#)


分线器 Splitter

名称	分线器（Splitter） 
行为	<p>分线器在一个多位值和这个值所能拆分出的几个单独子集之间创建对应关系。尽管它的名字是分线器，但它既可以将一个多位值拆分为数个更少位数的组件，也可以将数个组件合并为一个多位值——实际上它可以同时执行这两项操作。有关分线器的更完整描述，请参见《用户指南》的“分线器”部分。</p> <p>Logisim 在电路中传递值时会特别处理分线器：虽然所有其他组件都有计算过的延迟以模拟其行为，但分线器允许值瞬间传递。</p> <p>注：分线器（Splitter）是一个非标准术语，它是 Logisim 独有的。</p>
引脚	<p>为了区分分线器的几个连接点，我们将一侧的单个连接点称为其组合端，而将另一侧的多个连接点称之为其拆分端。</p> <p>组合端：（输入/输出位宽与“输入位宽”属性匹配）用于组合所有通过分线器的 bits。</p> <p>拆分端：（输入/输出，位宽基于“Bit x”属性得到）拆分端的数量由“扇出”属性指定，每个拆分端的编号至少为 0 且小于“扇出”属性值。对于每个拆分端，其编号所对应的 bit 会经过该拆分端；这些 bit 的顺序与它们在组合端的组合顺序相同。</p>
属性	<p>选择或添加组件时，按下数字“0”到“9”会更改其扇出属性，按下 Alt + 0 到 9 会同时更改扇出和输入位宽属性，按下方向键会更改其方向属性。</p> <ul style="list-style-type: none">● 方向（Facing）：拆分端相对于组合端的位置。● 扇出（Fan out）：拆分端的数量。● 输入位宽（Bit Width In）：组合端的位宽。● 外观（Appearance）：分线器在电路中有不同描绘方式。“左手（Left-handed）”选项（默认设置）从组合端向右上绘制主干，主干上每个拆分端对应位置引出一条标记线，形如大拇指朝下时手心朝外的左手。“右手（Right-handed）”选项相同，只是向右下绘制主干，形如大拇指朝上手心朝外的右手。“居中（Centered）”选项将主干水平居中，使其拆分端上下对称分布。“传统（Legacy）”选项将以多叉树的形式绘制拆分端，不带编号；此选项主要是为了与 2.7.0 之前的版本兼容，而 2.7.0 以前的版本该选项是分线器外观的唯一选项。● Bit x：组合端的位 x 对应的拆分端的编号。拆分端的编号从顶部的 0 开始（对于朝东或朝西的分线器），或从左侧/西部的 0 开始进行（对于朝北或朝南的分线器而言）。可以指定某个位不对应任何拆分端。一个单独的位无法对应多个拆分端。 <p>有时，您可以使用分线器的弹出菜单的某些选项（通常通过右键单击或 ctrl+单击分线器）来避免弄乱每个单独的 Bit x 属性。弹出菜单包括“升序分布（Distribute Ascending）”和“降序分布（Distribute Descending）”。“升序分布”选项分配所有位，以便每个拆分端从拆分端 0 开始接收相同数量的位。（如果拆分端的数量没有精确地划分为位的数量，则位将尽可能均匀地分布。）“降序分布”也一样，但从编号最高的拆分端开始。</p>
戳工具行为	无
文本工具行为	无


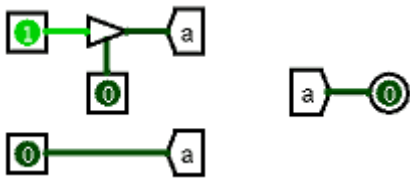
管脚 Pin

名称	管脚 (Pin) 
行为	<p>管脚用作电路的输出还是输入取决于其“输出/输入”属性。在绘制管脚时，Logisim 使用圆形或圆角矩形表示输出管脚，而输入管脚则使用方形或矩形表示。无论哪种情况，发送或接收的值的每个单个位都会显示在组件中（打印机视图中除外，此时组件只显示管脚的位宽）。</p> <p>管脚是一个与电路交互很方便的组件，并且 Logisim 的入门用户不需要用任何除上述外其他方式使用它们。但是，使用多个子电路构建电路的用户（如《用户指南》的“子电路”部分所述）还将使用管脚来指定电路和子电路之间的接口。尤其是当电路作为子电路被其他电路使用时，管脚在子电路中的位置将决定它在主电路中的布局。在这种电路中，由子电路组件上这些位置发送和接收的值与子电路布局中的管脚相关联。</p>
引脚	管脚组件只有一个引脚单元。如果该管脚是输出管脚，则组件向这个管脚输入值；如果该管脚是输入管脚，则这个管脚向组件输出一个值。
属性	<p>选择或添加组件时，按下 Alt + 0-9 会更改其“数据位宽”属性，按下方向键会更改其“方向”属性，按下 Alt + 方向键会更改其“标签位置”属性。</p> <ul style="list-style-type: none">● 方向 (Facing)：组件的输入/输出引脚所在的一侧。● 输出/输入 (Output?)：指定元件是输出管脚还是输入管脚。（请注意，如果管脚组件是一个输入管脚，那么作为电路内接口的管脚将向组件输出，反之亦然。）● 数据位宽 (Data Bits)：管脚包含的值的位数。● 三态 (Three-state?)：对于输入管脚，这配置了用户是否可以指示管脚发出未指定（即浮动）的值。该属性仅于用户界面处理；当电路布局用作子电路时，它对管脚的行为没有任何影响。对于输出管脚，该属性不起作用。● 上拉行为 (Pull Behavior)：对于输入管脚，该属性指定当作为输入（可能来自使用电路布局作为子电路的电路）接收浮动值时应如何处理。在“不变 (Unchanged)”的情况下，浮动值原封不动发送到布局中；使用“上拉 (pull up)”，在发送到电路布局之前，它们被转换为 1 值；使用“下拉 (pull down)”，它们在被发送到电路布局之前被转换为 0 值。● 标签 (Label)：与组件关联的标签内的文本。● 标签位置 (Label Location)：标签相对于组件的位置。● 标签字体 (Label Font)：用于呈现标签的字体。
截工具 行为	<p>单击输出管脚没有效果，但会显示管脚的属性。</p> <p>单击输入管脚将切换所单击的位的值。如果是三态管脚，则相应的位将在三态之间切换。</p> <p>但是，如果用户正在查看《用户指南》的“调试子电路”中描述的子电路状态部分，则管脚的值将固定为子电路从主电路接收到的某个值。如果不断开子电路和主电路之间的链接，用户将无法更改该值，Logisim 将提示用户验证是否确实需要断开该链接。</p>
文本工 具行为	允许编辑与组件关联的标签。

探针 Probe


名称	探针 (Probe) 
行为	探针是一种元件，顾名思义，它只显示电路中给定点的值。它本身并不与其他组件交互。 大多数情况下，探针元件与设置为输出管脚的管脚组件的功能相同。主要区别在于，如果电路用作子电路组件，则输出管脚将成为该接口的一部分，而探针则不会。探针同样也没有要设置的数据位宽属性：位宽是根据它在它的输入上所取到的值得到的。从图形上看，它们相似但边界略有不同：管脚的边界是黑色的，而探针的边界是灰色的。
引脚	探针元件只有一个引脚，它将作为探针的输入。该引脚接受的数据位宽是自适应的：探针将适应任何位宽的输入。
属性	选择或添加元件时，按下方向键会更改其“方向”属性。 <ul style="list-style-type: none">● 方向：组件的输入引脚所在的一侧。● 标签：与组件关联的标签内的文本。● 标签位置：标签相对于组件的位置。● 标签字体：用于呈现标签的字体。● 基数 (Radix)：显示值的进制（例如二进制、十进制或十六进制）。
戳工具行为	无
文本工具行为	允许编辑与元件关联的标签。

隧道 Tunnel


名称	隧道 (Tunnel) 
行为	<p>隧道的作用类似于导线，因为它将点绑定在一起，但与导线不同的是，点之间的连接没有明确绘制。当您需要连接电路中相距很远的点时，隧道这是很有用的，因为大量使用导线网络会使电路不太美观。下图说明了隧道是如何工作的：</p>  <p>在这里，所有三个隧道都具有相同的标签 a，因此这三个点代表同一个点。（如果其中一个隧道被标记为其他内容，如 b，则它将是另一组隧道的一部分）。顶部的受控缓冲区会输出不确定值，因为其下部输入为 0。这通常会导致来自受控缓冲区的导线为蓝色；但这里是深绿色的，因为浮动输出通过通道与底部管脚的 0 相结合。如果进入缓冲区的控制输入变为 1，则受控缓冲区将 1 送入通道，这将与底部管脚的 0 结合，从而产生错误值；在这种情况下，我们将看到红色导线连接所有三个隧道。</p>
引脚	隧道只有一个引脚，其位宽与隧道的“位宽”属性匹配。这个引脚既不是输入也不是输出——只是代表连接了匹配的隧道。

属性	当选择或添加组件时，按下 Alt + 0 到 9 更改其"位宽"属性，按下方向键更改其"方向"属性。 <ul style="list-style-type: none">● 方向：隧道指向的方向。● 数据位宽：隧道的位宽。● 标签：与隧道关联的标签内的文本。该隧道与标签完全相同的所有其他隧道相连。● 标签字体：用于呈现标签的字体。
戳工具行为	无
文本工具行为	允许编辑与隧道关联的标签。

上/下拉寄存器 Pull Resistor


名称	上/下拉寄存器 (Pull Resistor) 
行为	只有当连接到的点是浮动值 (Z) 时，该组件才有效。在这种情况下，寄存器将其连接的导线所带有的值拉向其"拉动方向"属性中指示的值。 如果它连接到一个多位值，那么浮动值中的每个位都会被拉向指定的值，而非浮动的位则保持不变。
引脚	电阻器只有一个用于输出的引脚，它的位宽来自于它连接的任何组件。
属性	选择或添加组件时，按下方向键会更改其"方向"属性。 <ul style="list-style-type: none">● 方向：组件相对于组件中心的方向。● 拉动方向 (Pull Direction)：指定应将浮动值拉动到的值。可以是 0、1 或 Error。
戳工具行为	无
文本工具行为	无

时钟 Clock


名称	时钟 (Clock) 
行为	只要通过"模拟 (Simulate)"菜单勾选"启用时钟 (Ticks Enabled)"，时钟就会按固定频率切换其输出值。(默认为禁用)。“Tick”是 Logisim 的时间单位；可以从"模拟"菜单的"计时频率 (Tick Frequency)"子菜单中选择计时发生的速度。 可以使用时钟的"高持续时间 (High Duration)"和"低持续时间 (Low Duration)"属性配置时钟周期。 请注意，Logisim 对时钟的模拟现实里基本不可能：在实际电路中，多个时钟会相互漂移，永远不会同步移动。但在 Logisim 中，所有时钟都以相同的速度计时。
引脚	一个时钟只有一个引脚，一个位宽为 1 的输出，其值将代表时钟的当前值。此接点的位置在"方向"属性中指定。无论何时启用计时，时钟的值都会按照原本的顺序切换，并且无论何时使用戳

	工具 Poke Tool 单击时钟其值都会切换。
属性	<p>选择或添加组件时，按下方向键会更改其“方向”属性。</p> <ul style="list-style-type: none">● 方向：组件的输出引脚所在的一侧。● 高（电平）持续时间：每个周期内时钟输出为 1 的时间长度。● 低持续时间：每个周期内时钟输出为 0 的时间长度。● 标签：与时钟组件关联的标签内的文本。● 标签位置：标签相对于组件的位置。● 标签字体：用于呈现标签的字体。
戳工具行为	单击时钟组件将立即切换其当前输出值。
文本工具行为	允许编辑与元件关联的标签。

常量 Constant








名称	常量 Constant 
行为	输出其“数值”属性中指定的值。
引脚	只有一个引脚，其位宽与“数据位宽”属性匹配。此接点的位置在“方向”属性中指定。组件会持续在此引脚上输出“数值”属性中指定的值。
属性	<p>选择或添加组件时，按下十六进制数字“0”到“9”和“a”到“f”会更改它的“数值”属性，按下 Alt + 0 到 9 会更改其位宽属性，按下方向键会更改其方向属性。</p> <ul style="list-style-type: none">● 方向：管脚相对于数据值的位置。● 位宽：传递到导线上的数据位宽。● 数值 (Value)：由组件输出的值，以十六进制形式显示。数值的位宽不能超过组件设定的位宽。
戳工具行为	无
文本工具行为	无

电源/接地 Power/Ground


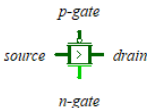
名称	电源/接地 Power/Ground 
行为	<p>将高电平（电源）或者低电平（地）接到导线上。对于由三角表示的符号，该值将为 1（如果“位宽”属性大于 1，则全为 1）。对于由三条缩短平行线表示的地符号，该值将为 0（如果“位宽”属性大于 1，则全为 0）。</p> <p>使用更通用的常量（Constant）组件可以实现相同的功能。选择接地和电源的唯一原因是它们是标准的电路符号。</p>
引脚	只有一个引脚，其位宽度与数据位属性匹配。该元件在该引脚上持续输出相同的值：对于接地符号，输出为全 0，对于电源符号，则输出为全 1。

属性	当选择或添加组件时，按下 Alt + 0 到 9 更改其“位宽”属性，按下方向键更改其“方向”属性。 <ul style="list-style-type: none">方向：箭头指向的方向。位宽：输出到导线上的值的位宽。
戳工具行为	无
文本工具行为	无


晶体管 Transistor

名称	晶体管 Transistor 																								
行为	<p>晶体管有两个输入端，称为栅极（gate）和源极（source），一个输出端称为漏极（drain）。图示时，源输入和漏输出由一个板连接；Logisim 绘制了箭头以指示从输入到输出的流动方向。栅极输入端被连接到一个与连接源极和漏极的板平行的板上。Logisim 支持两种类型的晶体管，其行为略有不同：</p> <ul style="list-style-type: none">● P 型晶体管中包含一个连接栅极输入端与其极板的圆圈。栅极为 0 时，电流可以从源极流向漏极。栅极为 1 时，源漏极断开，漏极处于悬浮（或高阻）状态。● N 型晶体管则没有这样的圆圈。栅极为 1 时，电流可以从源极流向漏极；栅极为 0 时，源漏极断开，漏极处于悬浮（或高阻）状态。 <p>下表总结了 P 型和 N 型晶体管的行为。</p> <table><tr><th colspan="2">P-type</th><th colspan="2">N-type</th></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td>gate</td><td>drain</td><td>gate</td><td>drain</td></tr><tr><td>0</td><td>source</td><td>0</td><td>Z</td></tr><tr><td>1</td><td>Z</td><td>1</td><td>source</td></tr><tr><td>X/Z</td><td>X*</td><td>X/Z</td><td>X*</td></tr></table> <p>源极、漏极的位宽由“位宽”属性设置，但栅极输入始终为 1bit。</p> <p>N 型晶体管的行为与受控缓冲区非常相似。主要区别是晶体管用于更基本的电路设计。</p>	P-type		N-type						gate	drain	gate	drain	0	source	0	Z	1	Z	1	source	X/Z	X*	X/Z	X*
P-type		N-type																							
																									
gate	drain	gate	drain																						
0	source	0	Z																						
1	Z	1	source																						
X/Z	X*	X/Z	X*																						
引脚	<p>假设组件朝东，栅极在上</p> <ul style="list-style-type: none">➤ 源极（输入，位宽由位宽属性设置）：由栅极触发时，元件的源输入将传输到输出。➤ 栅极（输入，位宽为 1）：对于 P 型晶体管，如果栅极值为 0，晶体管将导通；对于 N 型晶体管，如果栅极值为 1，晶体管将导通。➤ 漏极（输出，位宽由位宽属性设置）：输出结果由栅极、源极共同影响。																								
属性	<p>当选择或添加组件时，按下 Alt + 0 到 9 更改其"位宽"属性，按下方向键更改其"方向"属性。</p> <ul style="list-style-type: none">● 类型（Type）：确定晶体管是 P 型还是 N 型。● 方向（Facing）：组件的方向（其输出相对于其输入的方向）。● 门位置（Gate Location）：门输入的位置。● 位宽：组件输入和输出的位宽。																								
戳工具行为	无																								
文本工具行为	无																								

传输门 Transmission Gate

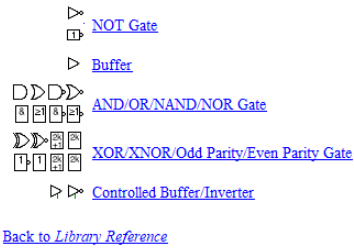
名称	传输门 Transmission Gate 																					
行为	<p>传输门有三个输入端，称为源、n 门和 p 门；有一个输出端，叫做漏（drain）。如图所示，源输入和漏输出由两个板连接；Logisim 绘制了箭头以指示从输入到输出的流动方向。两个门输入被画成线，连接到平行于连接源和漏的每个板的板上。p 门输入端的线有一个圆，而 n 门输入端则没有。</p> <div data-bbox="785 566 932 672"></div> <p>传输门只是两个互补晶体管的组合。事实上，在 Logisim 中，只要使用一个晶体管就可以实现相同的功能。然而，由于漏极电压带来的电学问题比 Logisim 能够模拟的更复杂，设计者有时更喜欢使用匹配的晶体管对。</p> <p>n 门和 p 门处的值按预计应当彼此相反。如果 p-gate 为 0，而 n-gate 为 1，则源处的值将传输到漏极。如果 p-gate 是 1 而 p-gate 为 0，则连接断开，因此漏极处的值保持浮动。在所有其他情况下，漏极接收错误输出，除非源极是浮动的，在这种情况下漏极也是浮动的。下表总结了这种行为。</p> <table data-bbox="785 972 932 1187"><thead><tr><th><i>p-gate</i></th><th><i>n-gate</i></th><th><i>drain</i></th></tr></thead><tbody><tr><td>0</td><td>0</td><td>X*</td></tr><tr><td>0</td><td>1</td><td><i>source</i></td></tr><tr><td>1</td><td>0</td><td>Z</td></tr><tr><td>1</td><td>1</td><td>X*</td></tr><tr><td>X/Z</td><td><i>any</i></td><td>X*</td></tr><tr><td><i>any</i></td><td>X/Z</td><td>X*</td></tr></tbody></table> <p>如果“位宽”属性大于 1，每个门输入仍然是一个 bit，但门值同时应用于源输入的每个位。</p>	<i>p-gate</i>	<i>n-gate</i>	<i>drain</i>	0	0	X*	0	1	<i>source</i>	1	0	Z	1	1	X*	X/Z	<i>any</i>	X*	<i>any</i>	X/Z	X*
<i>p-gate</i>	<i>n-gate</i>	<i>drain</i>																				
0	0	X*																				
0	1	<i>source</i>																				
1	0	Z																				
1	1	X*																				
X/Z	<i>any</i>	X*																				
<i>any</i>	X/Z	X*																				
引脚	<p>假设组件朝东，门线在上</p> <ul style="list-style-type: none">➤ 西侧（输入，位宽与数据位宽属性匹配）由 p 门和 n 门输入触发时，元件的源输入将传输到输出。➤ 北侧（输入，位宽为 1）组件的 p 门输入。➤ 南侧（输入，位宽为 1）组件的 n 门输入。➤ 东侧（输出，位宽与数据位宽属性匹配）组件的输出，如果 p-gate 为 0，n-gate 为 1，则与源输入匹配；如果 p-gater 为 1，n-gater 为 0，则它将是浮动的。对于 p-gate 和 n-gate 上的所有其他值，输出是错误值。																					
属性	<p>当选择或添加组件时，按下 Alt + 0 到 9 更改其“位宽”属性，按下方向键更改其“方向”属性。</p> <ul style="list-style-type: none">● 方向（Facing）：组件的方向（其输出相对于其输入的方向）。● 门位置（Gate Location）：门输入的位置。● 位宽：组件输入和输出的位宽。																					
戳工具行为	无																					
文本工具行为	无																					

位扩展器 Bit Extender


名称	位扩展器 Bit Extender 
行为	位扩展器将一个值转换为另一个位宽的值。如果将其转换为较小的位宽度，则只需将其截断以保留最低位。如果它被转换为一个更大的位宽，那么最低的位是相同的，并且可以选择额外的高位是什么：它们可以都是 0，都是 1，都与输入的符号位（最高的位）匹配，或者组件可以有一个额外的一位输入来确定这些其他位的属性。
引脚	<ul style="list-style-type: none">➤ 西侧（输入，来自“输入位宽”属性的位宽）需要转换的输入值。➤ 东侧（输出，“输出宽度”属性中的位宽）计算的输出值。➤ 北侧（输入，位宽为 1）指定输出中的附加位。仅当“扩展类型”属性为“输入”时，此引脚才可用。
属性	<p>当选择或添加组件时，按下数字 0 到 9 改变“输入位宽”属性，按下 Alt + 0 到 9 改变其“输出位宽”属性。</p> <ul style="list-style-type: none">● 输入位宽：需要转化的输入的位宽。● 输出位宽：转化得到的输出的位宽。● 扩展类型 (Extension Type)：假设输出位宽超过输入位宽，则此属性决定额外的输出位应该是什么。如果为 0 (Zero) 或 1 (One)，则附加位相应为 0 或 1。如果是符号位 (Sign)，则会采用额外的位来匹配输入中的最高顺位。如果为“输入 (Input)”，则组件在其北侧有第二个输入，其值用于额外的位。
戳工具行为	无
文本工具行为	无

第二节：逻辑门库 Gates library

逻辑门库包括各种简单组件, 所有这些组件都有一个单独的输出, 其值完全由当前输入决定。




非门 NOT Gate

名称	非门 NOT Gate 						
行为	<p>非门发出它接收到的任何输入的补码。非门的真值表如下。</p> <table><tr><td>x</td><td>out</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table> <p>如果输入未指定 (即浮动), 则输出也将未指定——除非“未定义时的门输出 (Gate Output When</p>	x	out	0	1	1	0
x	out						
0	1						
1	0						


	Undefined) "选项为"输入未定义时的错误 (Error for undefined inputs) ", 在这种情况下, 输出为错误。如果输入是一个错误值, 那么输出也将是。 多位非门将对其输入按位执行上述转换。
引脚	假定组件朝东 <ul style="list-style-type: none"> ● 西侧 (输入, 位宽由数据位宽属性确定): 组件的输入。 ● 东侧 (输出, 位宽由数据位宽属性确定): 输出, 其值是输入值按位取反。
属性	当选择或添加组件时, 按下 Alt + 0 到 9 更改其"数据位宽"属性, 按下方向键更改其"方向"属性。 <ul style="list-style-type: none"> ● 方向: 组件的方向 (其输出相对于其输入)。 ● 数据位宽: 组件输入和输出的位宽。 ● 尺寸 (Gate Size): 确定是绘制组件的较大版本还是较小版本。 ● 输出值: 指示应如何将假和真转换为输出值。默认情况下, false 由低电平 (0) 表示, true 由高电平 (1) 表示, 但可以用高阻抗 (浮动) 值代替其中一个值。这允许线或和线与连接, AND/OR/NAND/NOR Gate 文档也会提到。 ● 标签: 与门关联的标签内的文本。 ● 标签字体: 用于呈现标签的字体。
截工具行为	无
文本工具行为	允许编辑与逻辑门关联的标签。

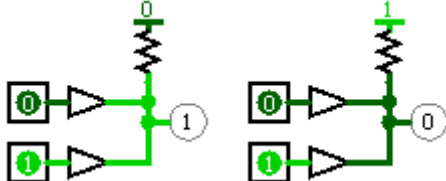
缓冲器 Buffer

名称	缓冲器 Buffer 
行为	<p>缓冲器会将其接收到的任何输入传递到输出。一位缓冲器的真值表如下。</p> <pre> x out 0 0 1 1 </pre> <p>如果输入未指定 (即浮动), 则输出也将未指定——除非提前将"未定义时的门输出"选项设置为"输入未定义时的错误", 在这种情况下, 输出为错误。如果输入是一个错误值, 那么输出将同样是一个错误值。</p> <p>缓冲器是 Logisim 所提供的最无用的门组件; 它在 Gates 库中的存在既是一个完整性问题 (保证每个可能的输入真值表的一个组件), 也是一个是否提供有用功能的问题。尽管如此, 确保值仅沿导线的一个方向传播有时还是很有用的。</p>
引脚	假定组件朝东 <ul style="list-style-type: none"> ● 西侧 (输入, 位宽由数据位宽属性确定): 组件的输入。 ● 东侧 (输出, 位宽由数据位宽属性确定): 输出, 始终与左侧的输入相匹配。
属性	当选择或添加组件时, 按下 Alt + 0 到 9 更改其"位宽"属性, 按下方向键更改其"方向"属性。 <ul style="list-style-type: none"> ● 方向: 组件的方向 (其输出相对于其输入)。 ● 数据位宽: 组件输入和输出的位宽。 ● 输出值: 指示应如何将假和真转换为输出值。默认情况下, false 由低电平 (0) 表示, true 由高电平 (1) 表示, 但可以用高阻抗 (浮动) 值代替其中一个值。允许线或和线与连接, 如 AND/OR/NAND/NOR 门 文档所示。 ● 标签: 与门关联的标签内的文本。



	● 标签字体：用于呈现标签的字体。
戳工具行为	无
文本工具行为	允许编辑与逻辑门关联的标签。

基本门 AND/OR/NAND/NOR Gate

名称	基本门 AND/OR/NAND/NOR Gate 																																																															
行为	<p>与门、或门、与非门、或非门分别计算输入的各自函数，并在输出上输出结果。</p> <p>默认情况下，任何未作任何连接的输入（如未连接导线）都将被忽略。因此可以插入一个 5 输入门，但只连接 2 个输入，它将作为一个 2 输入门工作；所以不必担心每次创建 gate 时都要配置输入的数量。（如果所有输入都未连接，则输出为错误值 X。）然而，一些用户更喜欢坚持所有输入都要连接，与现实世界的门相对应。你可以通过打开“项目->选项...”菜单项，选择“模拟”选项卡，然后为“未定义时的门输出”选择“输入未定义时的错误”来启用此行为。</p> <p>基本门的 2 输入真值表如下所示。（字母 X 表示错误值，字母 Z 表示浮动值。）</p> <table><tr><th colspan="2">AND</th><th colspan="2">OR</th></tr><tr><th>0</th><th>1</th><th>X/Z</th><th>0</th><th>1</th><th>X/Z</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>X</td></tr><tr><td>1</td><td>0</td><td>1</td><td>X</td><td>1</td><td>1</td><td>1</td></tr><tr><td>X/Z</td><td>0</td><td>X</td><td>X</td><td>X/Z</td><td>1</td><td>X</td></tr></table> <table><tr><th colspan="2">NAND</th><th colspan="2">NOR</th></tr><tr><th>0</th><th>1</th><th>X/Z</th><th>0</th><th>1</th><th>X/Z</th></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>X</td></tr><tr><td>1</td><td>1</td><td>0</td><td>X</td><td>1</td><td>0</td><td>0</td></tr><tr><td>X/Z</td><td>1</td><td>X</td><td>X</td><td>X/Z</td><td>X</td><td>0</td><td>X</td></tr></table> <p>简而言之，只要所有输入都是 0 或 1，这些组件就会按预期工作。如果其中一个输入既不是 0 也不是 1（它是浮动的或是错误值），那么组件会将其视为 0 和 1：如果两种结果输出都相同（就像与门有一个输入为 0，第二个输入有问题时也输出 0 的情况），那么这就是输出值；但是，如果输出的变化取决于它是 0 还是 1，那么输出就是错误值。</p> <p>每个门的多位版本将对其输入按位执行一位转换。</p>	AND		OR		0	1	X/Z	0	1	X/Z	0	0	0	0	0	1	X	1	0	1	X	1	1	1	X/Z	0	X	X	X/Z	1	X	NAND		NOR		0	1	X/Z	0	1	X/Z	0	1	1	1	0	1	X	1	1	0	X	1	0	0	X/Z	1	X	X	X/Z	X	0	X
AND		OR																																																														
0	1	X/Z	0	1	X/Z																																																											
0	0	0	0	0	1	X																																																										
1	0	1	X	1	1	1																																																										
X/Z	0	X	X	X/Z	1	X																																																										
NAND		NOR																																																														
0	1	X/Z	0	1	X/Z																																																											
0	1	1	1	0	1	X																																																										
1	1	0	X	1	0	0																																																										
X/Z	1	X	X	X/Z	X	0	X																																																									
引脚	<p>假定组件朝东</p> <p>➤ 西侧（输入，位宽由数据位宽属性决定）：组件的输入。输入由“输入数量”属性指定。请注意，如果使用的是成型的门电路，则或门与异或门的西侧将是弯曲的。尽管如此，输入引脚还是排成一行。Logisim 将绘制一些短线来说明这一点。在“打印机视图”中，除非将这些短线连接到导线，否则不会绘制这些短线。</p> <p>➤ 东侧（输出，位宽由数据位宽属性决定）：门的输出，其值根据上述电路输入计算。</p>																																																															
属性	<p>选择或添加组件时，按下数字“0”到“9”会更改其“输入数量”属性，按下 Alt + 0 到 9 会更改其数据位宽属性，按下方向键会更改其“方向”属性。</p> <ul style="list-style-type: none">● 方向：组件的方向（其输出相对于其输入）。● 数据位宽：组件输入和输出的位宽。● 尺寸：确定是绘制组件的更宽版本还是更窄版本。这不会影响由“输入数量”属性指定的输入数。然而，如果选择了成型门电路，则门将绘制为翼形，以容纳超出形状自然容纳范围的额外输入。● 输入数量：确定组件西侧的引脚数量。● 输出值：指示应如何将假和真转换为输出值。默认情况下，false 由低电平（0）表示，																																																															


	<p>true 由高电平（1）表示，但可以用高阻抗（浮动）值代替其中一个值。允许线或和线与连接，如下所示：在左侧，缓冲器的输出值为 floating/1，电阻器拉至 0，给出线或行为；在右侧，缓冲器的输出值为 0/floating，电阻器拉至 1，给出线与行为。</p>  <ul style="list-style-type: none">● 标签：与门关联的标签内的文本。● 标签字体：用于呈现标签的字体。● 负值 x：如果选择是，则输入在被送入门之前被取反。如果面朝东或西，则输入从上到下计数，如果面朝北或南，则从左到右计数。
截工具 行为	无
文本工 具行为	允许编辑与逻辑门关联的标签。

基本门 XOR/XNOR/Odd Parity/Even Parity Gate

名称	基本门 XOR/XNOR/Odd Parity/Even Parity Gate  																														
行为	<p>基本门的 2 输入真值表如下所示。</p> <table><tr><th>x</th><th>y</th><th>XOR</th><th>XNOR</th><th>Odd</th><th>Even</th></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table> <p>如您所见，奇校验门和异或门在两个输入端的行为相同；类似地，偶校验门和同或门的行为相同。但是，如果有两个以上的指定输入，则只有当正好有一个 1 输入时，异或门才会输出 1，而如果有奇数个 1 输入，奇偶校验门将输出 1。XNOR 门仅在没有一个 1 输入时输出 1，而偶校验门在偶数个 1 输入发出 1。异或和同或门包含一个名为“多输入行为”的属性，设置后可执行校验功能。</p>	x	y	XOR	XNOR	Odd	Even	0	0	0	1	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	1	0	1	0	1
x	y	XOR	XNOR	Odd	Even																										
0	0	0	1	0	1																										
0	1	1	0	1	0																										
1	0	1	0	1	0																										
1	1	0	1	0	1																										
引脚	<p>假定组件朝东</p> <p>➤ 西侧（输入，位宽由数据位宽属性决定）：组件的输入。输入由“输入数量”属性指定。请注意，如果使用的是成型的门电路，则或门与异或门的西侧将是弯曲的。尽管如此，输入引脚还是排成一行。Logisim 将绘制一些短线来说明这一点。在“打印机视图”中，除非将这些短线连接到导线，否则不会绘制这些短线。</p> <p>➤ 东侧（输出，位宽由数据位宽属性决定）：门的输出，其值根据上述电路输入计算。</p>																														
属性	<p>选择或添加组件时，按下数字“0”到“9”会更改其“输入数量”属性，按下 Alt + 0 到 9 会更改其数据位宽属性，按下方向键会更改其“方向”属性。</p> <ul style="list-style-type: none">● 方向：组件的方向（其输出相对于其输入）。● 数据位宽：组件输入和输出的位宽。● 尺寸：确定是绘制组件的更宽版本还是更窄版本。这不会影响由“输入数量”属性指定的																														

	<p>输入数。然而，如果选择了成型门电路，则门将绘制为翼形，以容纳超出形状自然容纳范围的额外输入。</p> <ul style="list-style-type: none">● 输入数量：确定组件西侧的引脚数量。● 输出值：指示应如何将假和真转换为输出值。默认情况下，<code>false</code> 由低电平（0）表示，<code>true</code> 由高电平（1）表示，但可以用高阻抗（浮动）值代替其中一个值。允许线和和线与连接。● 标签：与门关联的标签内的文本。● 标签字体：用于呈现标签的字体。● 多输入行为（异或同或专有）：当有三个及以上输入时，可以更改该属性来决定是否启动校验功能。
戳工具行为	无
文本工具行为	允许编辑与逻辑门关联的标签。

受控缓冲/反向器 Controlled Buffer/Inverter

名称	受控缓冲/反向器 Controlled Buffer/Inverter 
行为	<p>受控缓冲器和反向器（通常称为三状态缓冲器/逆变器）在南侧各有一个一位“控制”输入引脚。此处的值影响组件的行为：</p> <ul style="list-style-type: none">■ 当该引脚上的值为 1 时，组件的行为与相应不含控制单元的组件（缓冲器或逆变器（非门））的行为相同。■ 当值为 0 或未知（即浮动）时，组件的输出也是浮动的。■ 如果该值是错误值（例如，当两个冲突的值被一起输入时会发生），则输出是错误值。 <p>当您有一条导线（通常称为总线），其值应与几个组件之一的输出相匹配时，受控缓冲器非常有用。通过在每个组件输出和总线之间放置一个受控缓冲器，可以控制该组件的输出是否发送到总线。</p>
引脚	<p>假定组件朝东，控制线在南</p> <ul style="list-style-type: none">➤ 西侧（输入，位宽由数据位宽属性决定）：组件的输入。➤ 南侧（控制线，1 位）：控制单元，根据控制线的值决定输出。➤ 东侧（输出，位宽由数据位宽属性决定）：输出规则如行为所述。
属性	<p>当选择或添加组件时，按下 Alt + 0 到 9 更改其“数据位宽”属性，按下方向键更改其“方向”属性。</p> <ul style="list-style-type: none">● 方向：组件的方向（其输出相对于其输入）。● 数据位宽：组件输入和输出的位宽。● 尺寸：（仅限受控逆变器）确定是绘制组件的较大版本还是较小版本。● 控制线位置：控制线的位置，假设我们正面对输出：如果组件朝东且右手系，则控制线在南边；但如果它是左手系，控制线就在北边。● 标签：与门关联的标签内的文本。● 标签字体：用于呈现标签的字体。
戳工具行为	无
文本工具行为	允许编辑与逻辑门关联的标签。


具行为	
-----	--

第三节：复用器库 Plexers library


Plexers 库包括控制组件。与逻辑门库的组件一样，所有组件都是组合的，但它们的用途通常是用于路由值。

-  [Multiplexer](#)
-  [Demultiplexer](#)
-  [Decoder](#)
-  [Priority Encoder](#)
-  [Bit Selector](#)


多路选择器 Multiplexer

名称	多路选择器 Multiplexer 
行为	通过南侧的输入选定将西侧的哪一个输入作为东侧的输出。
引脚	<p>假设组件面向东，选择线在底部左侧。</p> <ul style="list-style-type: none"> ➤ 西侧（输入，位宽由数据位宽属性决定）：输入数个数据值，其中一个将发送到输出。每个输入数据值都有编号，从北面的 0 开始。 ➤ 东侧（输出，位宽由数据位宽属性决定）：输出值将与西边的输入值相匹配，该值的编号与当前通过南边的选择输入接收到的值相同。如果选择输入包含任何未指定（即浮动）位，则输出完全浮动。 ➤ 南侧左边：用灰色圆圈表示（输入，位宽由“选择位宽”属性决定），即选择输入。此输入的值确定要发送到东侧输出的西侧输入。 ➤ 南侧右边（输入，位宽度 1）：使能线。为 0 时，多路选择器的输出由所有浮动位组成，而与数据输入和选择输入无关。
属性	<p>选择或添加组件时，按下数字“1”到“4”会更改其“选择位宽”属性，按下 AI + 0 到 9 会更改其数据位宽属性，按下方向键会更改其“方向”属性。</p> <ul style="list-style-type: none"> ● 方向：组件的方向（其输出相对于其输入）。 ● 选择位置：相对于元件的选择线和使能线的位置。 ● 选择位宽：组件的选择输入的位宽，设为 x。多路选择器的输入数量为 2^x。 ● 数据位宽：通过多路选择器发送的数据的位宽。 ● 禁用时的输出（Disabled Output）：指定禁用组件时（即使能线为 0 时），输出的每个位应该是什么。选项包括零和浮动；在后一种情况下，输出能有效地与任何其他端口断开。 ● Include Enable?：当该属性为“是”时，组件具有启用输入。该属性主要用于支持使用未提供启用输入的旧版本 Logisim 构建的电路。
戳工具行为	无
文本工具行为	无

多路分配器 Demultiplexer


名称	多路分配器 Demultiplexer 
行为	通过南侧的输入选定将西侧的输入发送到东侧的哪一个输出上
引脚	<p>假设组件面向东，选择线在底部左侧。</p> <ul style="list-style-type: none">➤ 西侧（输入，位宽由数据位宽属性决定）：输入一个将被发送到东侧的值。➤ 东侧（输出，位宽由数据位宽属性决定）：输出从北侧的 0 开始编号。其中一个与南侧选择输入对应的输出将得到西侧输入的值。若无匹配，则全部输出 0 或浮动值，取决于“三态”属性。如果选择输入也有不确定位，则输出全为浮动值。➤ 南侧右边：用灰色圆圈表示（输入，位宽由“选择位宽”属性决定），即选择输入。此输入的值确定得到西侧输入的东侧输出。➤ 南侧左边（输入，位宽度 1）：使能线。为 0 时，多路分配器的所有输出由所有浮动位组成，而与数据输入和选择输入无关。
属性	<p>选择或添加组件时，按下数字“1”到“4”会更改其“选择位宽”属性，按下 AI + 0 到 9 会更改其数据位宽属性，按下方向键会更改其“方向”属性。</p> <ul style="list-style-type: none">● 方向：组件的方向（其输出相对于其输入）。● 选择位置：相对于元件的选择线和使能线的位置。● 选择位宽：组件的选择输入的位宽，设为 x。多路分配器的输出数量为 2^x。● 数据位宽：通过多路分配器发送的数据的位宽。● 禁用时的输出（Disabled Output）：指定禁用组件时（即当使能线为 0 时），输出的每个位应该是什么。选项包括零和浮动；在后一种情况下，输出能有效地与任何其他端口断开。● Include Enable?：当该属性为“是”时，组件具有启用输入。该属性主要用于支持使用未提供启用输入的旧版本 Logisim 构建的电路。
戳工具行为	无
文本工具行为	无

解码器 Decoder

名称	解码器 Decoder 
行为	仅在一个输出端口输出 1，哪一个端口取决于当前南侧的输入值。没有西侧输出
引脚	<p>假设组件面向东，选择线在底部左侧。</p> <ul style="list-style-type: none">➤ 东侧（输出，位宽为 1）：输出从北侧的 0 开始编号。其中一个与南侧选择输入对应的输出将输出 1。若未能匹配，则输出 0 或浮动值，取决于“三态”属性。如果选择输入也有不确定位，则输出全为浮动值。➤ 南侧右边：用灰色圆圈表示（输入，位宽由“选择位宽”属性决定），即选择输入。此输入的值确定得到 1 的东侧输出。➤ 南侧左边（输入，位宽度 1）：使能线。为 0 时，解码器的所有输出由所有浮动位组成，而与数据输入和选择输入无关。
属性	选择或添加组件时，按下数字“1”到“4”会更改其“选择位宽”属性，按下 AI + 0 到 9 会更改其数据


	<p>位宽属性，按下方向键会更改其“方向”属性。</p> <ul style="list-style-type: none"> ● 方向：组件的方向（其输出相对于其输入）。 ● 选择位置：相对于元件的选择线和使能线的位置。 ● 选择位宽：组件的选择输入的位宽，设为 x。解码器的输出数量为 2^x。 ● 数据位宽：通过解码器发送的数据的位宽。 ● 禁用时的输出（Disabled Output）：指定禁用组件时（即当使能线为 0 时），输出的每个位应该是什么。选项包括零和浮动；在后一种情况下，输出能有效地与任何其他端口断开。 <p>Include Enable?：当该属性为“是”时，组件具有启用输入。该属性主要用于支持使用未提供启用输入的旧版本 Logisim 构建的电路。</p>
戳工具行为	无
文本工具行为	无

优先编码器 Priority Encoder

名称	优先编码器 Priority Encoder 
行为	<p>组件的西侧有许多输入，第一个编号为 0，依次往下。该组件记录值为 1 的输入的编号，并输出最高的编号。例如，如果 0、2、5 和 6 号输入均为 1，则优先编码器输出值 110（6 的 2 进制）。如果没有输入为 1，或组件被禁用，则优先级解码器的输出为浮动值。</p> <p>优先编码器的设计使许多编码器可以菊链式（daisy-chained）连接，以容纳额外的输入。该组件包括使能输入和使能输出。每当使能输入为 0 时，组件将被禁用，输出将全部为浮动值。只要组件已启用，且输入均不为 1，则使能输出为 1。因此，您可以采用两个优先编码器，将第一个编码器的使能输出连接到第二个编码器的使能输入：如果第一个的任何输入为 1，那么第二个将被禁用，因此其输出都将是浮动的。但是，如果第一个索引输入中没有一个是 1，那么它的输出将全部是浮动值，第二个优先编码器将被启用。</p> <p>每当启用优先编码器并在其中一个输入上找到 1 时，优先编码器的一个附加输出为 1。当将优先编码器链接在一起时，此输出可用于识别触发了哪个编码器。</p>
引脚	<p>假定组件朝东</p> <ul style="list-style-type: none"> ➤ 西侧，可变数字（输入，位宽度 1）输入值，从西侧上方的 0 开始编号。 ➤ 东侧，上方引脚（输出，位宽由选择位宽决定）输出值为 1 的输入中的最高编号，如果没有输入为 1，或如果通过启用输入禁用组件，则全为浮动值。 ➤ 东侧，下引脚（输出，位宽度 1）组信号：如果组件已启用，并且至少有一个索引输入的值为 1，则输出为 1；否则，此输出为 0。 ➤ 南侧（输入，位宽度 1）使能输入：如果为 0，则组件被禁用；否则将启用该组件。 ➤ 北侧（输出，位宽度 1）使能输出：如果此组件已启用，且索引输入均不为 1，则为 1；否则输出为 0。
属性	<p>选择或添加组件时，按下数字“1”到“4”会更改其“选择位宽”属性，按下方向键会更改其“方向”属性。</p> <ul style="list-style-type: none"> ● 方向：组件的方向（其输出相对于其输入）： ● 选择位宽：组件主输出的位宽，设为 x。优先编码器的输入数为 2^x。 ● 禁用的输出：指定禁用组件时（即，当使能引脚为 0 时），输出的每个位应该是什么。选

	项包括零和浮动；在后一种情况下，输出有效地与任何其他端口断开。
戳工具行为	无
文本工具行为	无

位选择器 Bit Selector


名称	位选择器 Bit Selector 
行为	<p>给定一个多位输入，把它分成几个位数相等的组（从最低顺序的位开始），并输出由选择输入选择的组。</p> <p>例如，如果我们有一个八位输入 01010101，有一个三位输出，那么组 0 将是最低顺序的三位 101，组 1 将是接下来的三位，010，组 2 将是接下来三位 001；如果选择输入为 3，则输出为 000。</p>
引脚	<p>假定组件朝东</p> <ul style="list-style-type: none"> ➤ 西侧（输入，位宽与数据位宽属性匹配）：输入需拆分的数值。 ➤ 东侧（输出，位宽与输出位宽属性匹配）：数据值中的一组位，由选择输入选择。 ➤ 南边（输入，位宽是数据位宽和输出位宽的商，向上取整）：选择输入，确定应发送到输出的位组。
属性	<p>选择或添加组件时，按下数字“0”到“9”会更改其“输出位宽”属性，按下 Alt + 0 到 Alt + 9 会更改其数据位宽属性，按下方向键会更改其“方向”属性。</p> <ul style="list-style-type: none"> ● 方向：组件的方向（其输出相对于其输入）。 ● 数据位宽：组件数据输入的位宽。 ● 输出位宽：组件输出的位宽。
戳工具行为	无
文本工具行为	无

第四节：运算器库 Arithmetic library


算术库包括对无符号数和二进制补码数执行算术运算的组合逻辑组件。

-  [Adder](#)
-  [Subtractor](#)
-  [Multiplier](#)
-  [Divider](#)
-  [Negator](#)
-  [Comparator](#)
-  [Shifter](#)
-  [Bit Adder](#)
-  [Bit Finder](#)

加法器 Adder


名称	加法器 Adder 
行为	<p>该组件将通过西侧两个输入的值相加，并在东侧输出总和。该组件的设计使其可以与其他加法器级联，以提供比单个加法器更多的位：进位输入提供一个要加到总和中的一位值，进位输出提供一个可以发送到另一个加法器的一位溢出值。</p> <p>如果任何一个加数包含一些浮动位或错误位，则组件将执行部分加法。也就是说，它将计算尽可能多的低阶位。但在浮动位或错误位之上，结果将具有浮动位或错误位。</p>
引脚	<ul style="list-style-type: none">➤ 西侧上方（输入，位宽与数据位宽属性匹配）：要相加的两个值之一。➤ 西侧下方（输入，位宽与数据位宽属性匹配）：要相加的两个值中的另一个。➤ 北侧，标记为 c in（输入，位宽度 1）：要加到总和中的进位值。如果值未知（即浮动），则假定为 0。➤ 东侧（输出，位宽与数据位宽属性匹配）：较低的位是来自西侧的两个值之和的位，最后加上 c in 位。➤ 南侧，标记为 c out（输出，位宽度 1）：为总和计算的进位。如果值加在一起作为无符号值产生数据位宽能容纳的结果，则该位将为 0；否则，它将为 1。
属性	<p>选择或添加组件时，按下 Alt + 0 到 9 会更改其"数据位宽"属性。</p> <ul style="list-style-type: none">● 数据位宽：要相加的值和结果的位宽。
戳工具行为	无
文本工具行为	无

减法器 Subtractor


名称	减法器 Subtractor 
行为	<p>该组件减去通过西侧输入（上部减去下部）的值，并在东侧输出差值。该组件的设计使其可以与其他减法器级联，以提供比使用单个减法器所能提供的减法位数更多的减法位：借入输入提供一个从差值中借出的一位值（如果指定了借入输入），借出输出指示组件是否需要借入高位来完成减法而无下溢（假设无符号减法）。</p> <p>在内部，减法器只是对减数执行按位取非，并将其与借入输入取非一起加到被减数。如果任一操作数包含一些浮动位或错误位，则组件将执行部分减法。也就是说，它将计算尽可能多的低阶位。但在浮动位或错误位之上，结果将具有浮动为和错误位。</p>
引脚	<ul style="list-style-type: none">➤ 西侧上方（输入，位宽与数据位宽属性匹配）：被减数。➤ 西侧下方（输入，位宽与数据位宽属性匹配）：减数。➤ 北侧，标记为 b in（输入，位宽度 1）：借入输入，即被借位。如果值未知（即浮动），则假定为 0。➤ 东侧（输出，位宽与数据位宽属性匹配）：较低的位是来自西侧的两个值之差的位，最后减去 c in 位。➤ 南侧，标记为 b out（输出，位宽度 1）：为计算出的借位。如果值加在一起作为无符号值产生负值，则该位将为 1；否则，它将为 0。

属性	选择或添加组件时，按下 Alt + 0 到 9 会更改其"数据位宽"属性。 数据位宽：要相加的值和结果的位宽。
戳工具行为	无
文本工具行为	无

乘法器 Multiplier


名称	乘法器 Multiplier 
行为	与加法器基本相同。值得注意的是，进位值可以提供多位。
引脚	<ul style="list-style-type: none">➤ 西侧上方（输入，位宽与数据位宽属性匹配）：要相乘的两个值之一。➤ 西侧下方（输入，位宽与数据位宽属性匹配）：要相乘的两个值中的另一个。➤ 北侧，标记为 c in（输入，位宽与数据位宽属性匹配）：要加到积中的进位值。如果值未知（即浮动），则假定为 0。➤ 东侧（输出，位宽与数据位宽属性匹配）：较低的位是来自西侧的两个值之积的位，最后加上 c in 位。 南侧，标记为 c out （输出，位宽与数据位宽属性匹配）：积的高位
属性	选择或添加组件时，按下 Alt + 0 到 9 会更改其"数据位宽"属性。 <ul style="list-style-type: none">● 数据位宽：要相加的值和结果的位宽。
戳工具行为	无
文本工具行为	无

除法器 Divider


名称	除法器 Divider 
行为	大致与减法器相同。值得注意的是，如果除数为 0，则不执行除法（即，假设除数为 1）。除法器实际上执行无符号除法。也就是说，余数总是介于 0 和除数-1 之间。商总是一个整数，因此 ✓ 商*除数+余数=被除数。 但是，如果数据位宽不适合商，则指指示较低位。该组件不提供任何访问高位数据位的方法。如果其中一个操作数包含一些浮动位或错误位，那么组件的输出要么是完全浮动的，要么是完全错误的值。
引脚	<ul style="list-style-type: none">➤ 西侧北方：（输入，位宽与数据位宽属性匹配）：被除数的低位数据。➤ 西侧南方：（输入，位宽与数据位宽属性匹配）：除数（即除法的第二个操作数）。➤ 北侧：（输入，位宽与数据位宽属性匹配）：被除数的高位数据。➤ 东侧：（输出，位宽度与数据位宽属性匹配）：商的低位数据，如上所述。➤ 南边，标记为 rem（输出，位宽与数据位宽属性匹配）：余数。该值始终介于 0 和除数-1 之

	间。
属性	选择或添加组件时，按下 Alt + 0 到 9 会更改其"数据位宽"属性。 <ul style="list-style-type: none">数据位宽：要相加的值和结果的位宽。
戳工具行为	无
文本工具行为	无

取反器 Negator


名称	取反器 Negator 
行为	计算输入二进制补码的求反。这种求反是通过保持值为 1 的最低位，并对其上的所有位进行取反来实现的。 如果要求反的值恰好是最小负值，那么它的求反（不能用补码形式表示）仍然是最小负数。
引脚	<ul style="list-style-type: none">西侧（输入，位宽与数据位宽属性匹配）：要求发的值。东侧，标记为-x（输出，位宽与数据位宽属性匹配）：输入的相反数。然而，如果输入恰好是数据位中可表示的最小负值，则输出与输入匹配。
属性	选择或添加组件时，按下 Alt + 0 到 9 会更改其"数据位宽"属性。 <ul style="list-style-type: none">数据位宽：组件输入和输出的位宽度。
戳工具行为	无
文本工具行为	无

比较器 Comparator

名称	比较器 Comparator 
行为	根据"数字类型"属性，将两个值作为无符号值或两个补码值进行比较。通常，三个输出中一个输出为 1，其他两个输出为 0。 比较从每个数字的最高有效位开始，并行向下递减，直到找到两个值不一致的位置。但是，如果在下降过程中遇到错误值或浮动值，则所有输出将与该错误或浮动值匹配。
引脚	<ul style="list-style-type: none">西侧，北方（输入，位宽与数据位宽属性匹配）：要比较的两个值中的第一个。西侧，南方（输入，位宽与数据位宽属性匹配）：要比较的两个值中的第二个。东侧，标记为>（输出，位宽度 1）：如果第一个输入大于第二个输入，则为 1；如果第一个输出小于或等于第二个输出，则为 0。东侧，标记为=（输出，位宽度 1）：如果第一个输入与第二个输入相等，则为 1；如果第一个输出不等于第二个输出，则为 0。东侧，标记为<（输出，位宽度 1）：如果第一个输入小于第二个输入，则为 1；如果第一个输出大于或等于第二个输出，则为 0。
属性	选择或添加组件时，按下 Alt + 0 到 9 会更改其"数据位宽"属性。

	<ul style="list-style-type: none">● 数据位宽：组件输入的位宽。● 数字类型（Numeric Type）：选择是无符号还是二进制补码比较。
戳工具行为	无
文本工具行为	无

移位器 Shifter


名称	移位器 Shifter 
行为	<p>此组件包括两个输入，数据和距离，它有一个输出，这是按距离移动数据的结果。数据和输出的位数相同。该组件支持以下移位类型：</p> <ul style="list-style-type: none">✓ 逻辑左：数据中的所有位都向左移位 <code>dist</code> 距离，右部用 0 填充。例如，11001011 逻辑左移两次为 00101100。（前两位丢失）✓ 逻辑右：数据中的所有位都向右移位 <code>dist</code> 距离，空缺位置用 0 填充。例如，11001011 逻辑右移两次为 00110010。（后两位丢失）✓ 算术右移：数据中的所有位都向右移位 <code>dist</code> 距离，空缺位置填充数据中任何最高位的重复。例如，11001011 算术右移两次为 11110010。✓ 左旋：数据中的所有位都向左移位 <code>dist</code> 距离，被移出的位排在右部。例如，11001011 向左旋转两次即为 00101111。✓ 右旋：数据中的所有位向右移位 <code>dist</code> 距离，被移出的位排在左部。例如，11001011 向右旋转两次即为 11110010。 <p>请注意，如果 <code>dist</code> 包含任何浮点值或错误输入，那么输出完全由错误值组成，因为无法猜测输入的偏移量。</p>
引脚	<ul style="list-style-type: none">➤ 西侧北方（输入，位宽与数据位宽属性匹配）：要移位的值。➤ 西侧南方（输入，位宽计算如下）：移位数据输入的位数。该输入应具有尽可能多的位，以指示从 0 到小于数据位 1 的任何移位距离；也就是说，它应该是数据位数以 2 为底的对数的上限。例如，如果数据位数为 8，则此输入需要 3 位；但如果是 9，则需要 4 位。➤ 东侧（输出，位宽与数据位宽属性匹配）：按输入距离移动输入值的结果。
属性	<p>选择或添加组件时，按下 Alt + 0 到 9 会更改其“数据位宽”属性。</p> <ul style="list-style-type: none">● 数据位宽：数据输入和输出的位宽度。● 移位类型：上述五种可能的移位类型之一（逻辑左移、逻辑右移、算术右移、左旋、右旋）。
戳工具行为	无
文本工具行为	无

位加法器 Bit Adder

名称	位加法器 Bit Adder 
----	--

行为	<p>组件计算输入所有位的和（亦即输入中共有多少 1）。</p> <p>如果任何输入位为浮动值或错误值，则输出中奖包含与可能输出范围相对应的错误位，具体取决于这些浮动/错误值是计为零还是计为一。例如，如果 14 位输入为 111x10110x1101，则输出必须至少为 9（如果 x 被解释为 0），最多为 11（如果 x 解释为 1）。因此，输出将是 10EE：由于 9 和 11 之间的所有整数的前两位都是 1 和 0，因此上两位将是 1 和零，但下两位是 EE，因为 9 和 11 间的整数在这些位中变化。</p>
引脚	<ul style="list-style-type: none"> ➤ 西侧（输入，位宽与数据位宽属性匹配）：需要计算共有多少个 1 的输入。输入数量基于“输入数”属性。 ➤ 东侧（输出，按如下所述计算的位宽度）：输出的位宽是存储最大可能值的最小位数（它是“数据位宽”属性和“输入数”属性的乘积）。
属性	<p>选择或添加组件时，按下数字“0”到“9”会更改其“输入数”属性，按下 Alt + 0 到 9 会更改其数据位宽属性。</p> <ul style="list-style-type: none"> ● 数据位宽：输入的位宽。 ● 输入数：共多少输入。
戳工具行为	无
文本工具行为	无

位索引器 Bit Finder

名称	位索引器 Bit Finder 										
行为	<p>该组件接受一个多位输入并确定其中一个位的编号，其中编号是通过从 0 开始计数作为最低顺序位来计算的。它计算的确切编号取决于 Type 属性，如下表中 8 位样本输入 11010100 的示例所示。</p> <table> <tr> <th>Type</th><th>Output for 11010100</th></tr> <tr> <td>Lowest-order 1</td><td>2</td></tr> <tr> <td>Highest-order 1</td><td>7</td></tr> <tr> <td>Lowest-order 0</td><td>0</td></tr> <tr> <td>Highest-order 0</td><td>5</td></tr> </table> <p>对于最低阶 1，输出为 2，因为如果对最低阶位从 0 开始的位进行索引，您将在编号 2 处找到第一个 1。（0 和 1 处的位都为 0。）对于最高阶 1，输出为 7，因为最高阶 1 位在比编号 7 处（同样从最低阶位算起为 0）。</p> <p>组件在南边的输出指示是否找到了所需的位。在上述涉及输入 11010100 的示例中，南部输出在所有情况下都是 1。但如果输入为 00000000，组件将查找最低阶的 1，那么南边输出将为 0，东边的输出也将为 0。</p> <p>如果在搜索所需值时，发现一个既不是 0 也不是 1 的值（该位可能是浮动的或错误值），则两个输出都将完全由错误位组成。注意，只有在找到所需位之前遇到问题位时才会发生这种情况：对于输入 x1010100，如果需要最低阶 1，输出仍然是 2；但是，如果组件的类型指示搜索最高阶 1 或最高阶 0，我们会得到错误值，因为在高于最高阶 0 或最高阶 1 的高阶位中存在错误位。</p>	Type	Output for 11010100	Lowest-order 1	2	Highest-order 1	7	Lowest-order 0	0	Highest-order 0	5
Type	Output for 11010100										
Lowest-order 1	2										
Highest-order 1	7										
Lowest-order 0	0										
Highest-order 0	5										
引脚	<ul style="list-style-type: none"> ➤ 西侧（输入，位宽与数据位宽属性匹配）：要搜索所需位的多位输入。 ➤ 东侧（输出，按如下所述计算的位宽）：所需位的编号，从 0 开始计算最低顺序位。位宽是存储最大可能编号的最小位数，它比“数据位宽”属性的值小 1。 										





	➤ 南侧（输出，位宽度 1）：如果找到所需位，则为 1；如果所有输入位都是所需位的倒数，则为 0；如果在所需位之前找到非 0、非 1 值，则为错误值。
属性	选择或添加组件时，按下 Alt + 0 到 9 会更改其"数据位宽"属性。 <ul style="list-style-type: none">● 数据位宽：输入的位宽。● 类型：指示要搜索的位——最低阶 0、最高阶 0、最低阶 1 或最高阶 1。
截工具行为	无
文本工具行为	无

第五节：存储器库 Memory library

内存库包括记忆信息的组件。




触发器 D/T/J-K/S-R Flip-Flop

名称	触发器 D/T/J-K/S-R Flip-Flop    																																																							
行为	每个触发器存储一位数据，该数据通过东侧的 Q 输出发送。通常，该值可以通过西侧的输入进行控制。特别地，当每个触发器上用三角形标记的时钟输入从 0 上升到 1（或按设置）时，该值会发生变化；在这个上升沿上，值根据下表变化。																																																							
	<table><tr><th colspan="2">D Flip-Flop</th><th>T Flip-Flop</th><th>J-K Flip-Flop</th><th>S-R Flip-Flop</th></tr><tr><th>D</th><th>Q</th><th>T</th><th>Q</th><th>J</th><th>K</th><th>Q</th><th>S</th><th>R</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>0</td><td>Q</td><td>0</td><td>0</td><td>Q</td><td>0</td><td>0</td><td>Q</td></tr><tr><td>1</td><td>1</td><td>1</td><td>Q'</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td></td><td></td><td></td><td></td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr><tr><td></td><td></td><td></td><td></td><td>1</td><td>1</td><td>Q'</td><td>1</td><td>1</td><td>??</td></tr></table>	D Flip-Flop		T Flip-Flop	J-K Flip-Flop	S-R Flip-Flop	D	Q	T	Q	J	K	Q	S	R	Q	0	0	0	Q	0	0	Q	0	0	Q	1	1	1	Q'	0	1	0	0	1	0					1	0	1	1	0	1					1	1	Q'	1	1	??
	D Flip-Flop		T Flip-Flop	J-K Flip-Flop	S-R Flip-Flop																																																			
	D	Q	T	Q	J	K	Q	S	R	Q																																														
	0	0	0	Q	0	0	Q	0	0	Q																																														
	1	1	1	Q'	0	1	0	0	1	0																																														
					1	0	1	1	0	1																																														
					1	1	Q'	1	1	??																																														
	用语言描述如下（最好参见课本）：																																																							
	✓ D 触发器：当时钟触发时，触发器记忆的值在那一刻成为 D 输入（数据）的值。																																																							
✓ T 触发器：当时钟触发时，触发器记忆的值会反转或保持不变，这取决于 T 输入（Toggle）是 1 还是 0。																																																								
✓ J-K 触发器：当时钟触发时，如果 J 和 K 输入均为 1，触发器记忆的值将切换，如果均为 0，则值保持不变；如果它们不同，则如果 J（Jump）输入为 1，则值变为 1；如果 K（Kill）输入为 1，则值为 0。																																																								
✓ S-R 触发器：当时钟触发时，如果 R 和 S 都为 0，触发器记住的值保持不变；如果 R 输入（复位）为 1，则变为 0；如果 S 输入（设置）为 1，则变为 1。如果两个输入都为 1，触																																																								


	<p>发器将发生未知行为。（在 Logisim 中，触发器中的值保持不变。）</p> <p>默认情况下，时钟在上升沿触发，即当时钟输入从 0 变为 1 时。但是，触发器属性允许它变为下降沿（时钟输入从 1 变为 0 时）、高电平（时钟输入为 1 的持续时间）或低电平（时钟输入为 0 的持续时间时）。电平触发器选项不适用于 T 和 J-K 触发器，因为触发器在被告知切换不确定的时间段时会表现出不可预测的行为。</p>
引脚	<ul style="list-style-type: none">➤ 西侧，用三角形标记（输入，位宽度 1）：时钟输入。在该输入值从 0 切换到 1（上升沿）的瞬间，该值将根据西侧的其他输入进行更新。只要它保持为 0 或 1，西边缘上的其他输入就不起作用。➤ 西侧，其他带标签的引脚（输入，位宽度 1）：这些输入控制触发器值在时钟上升沿期间的变化。它们的确切行为取决于触发器；上表总结了它们的行为。➤ 东侧，标记为 Q，北端（输出，位宽度 1）：输出触发器当前存储的值。➤ 东侧，南端（输出，位宽度 1）：输出触发器当前存储的值的补码。➤ 南侧，东端（输入，位宽度 1）：异步复位。当值为 0 或未定义时，此输入无效。只要它是 1，触发器的值就固定为 0。这是异步发生的，即与当前时钟输入值无关。只要这是 1，其他输入就没有任何影响。➤ 南侧，中间（输入，位宽度 1）：使能。当此值为 0 时，时钟触发器将被忽略。当前位继续出现在输出上。当该输入为 1 或未定义时，时钟触发器启用。➤ 南边，西端（输入，位宽度 1）：异步设置。如果为 0 或未定义，则此输入无效。当为 1 时，触发器的值固定为 1。这是异步发生的，即与当前时钟输入值无关。只要该输入为 1，其他输入就没有作用，但异步复位输入除外，异步复位输入具有优先权。
属性	<ul style="list-style-type: none">● 触发 (Trigger)：配置如何解释时钟输入。上升沿值表示触发器应在时钟从 0 上升到 1 的瞬间更新其值。下降沿值表示它应在时钟由 1 下降到 0 的瞬间更新。高电平值表示触发器在时钟输入为 1 时应持续更新。低电平值表示当时钟输入为 0 时应持续更新。请注意，后两个选项对于 T 和 J-K 触发器不可用。● 标签：与触发器关联的标签内的文本。● 标签字体：用于呈现标签的字体。
戳工具行为	除非异步设置/复位输入锁定当前触发器的值，否则使用戳工具 Poke Tool 点击触发器可切换存储在触发器中的位。
文本工具行为	允许编辑与元件关联的标签。

寄存器 Register

名称	寄存器 Register 
行为	<p>寄存器存储单个多位值，该值以十六进制显示在其矩形内，并在其 Q 输出上发送。当时钟输入（由南边的三角形表示）满足触发条件，寄存器中存储的值会立即变为 D 输入的值。触发条件自行设置。</p> <p>复位输入将寄存器的值异步复位为 0（全部为零）；也就是说，只要复位输入为 1，该值就固定为 0，而与时钟输入无关。</p>
引脚	<ul style="list-style-type: none">➤ 东侧，标记为 Q（输出，位宽与数据位宽属性匹配）：输出寄存器当前存储的值。➤ 西侧，标记为 D（输入，位宽与数据位宽属性匹配）：数据输入。在时钟值从 0 上升到 1 的瞬间（上升沿触发），寄存器的值变为此时 D 输入的值。


	<ul style="list-style-type: none">➤ 西侧，标记为 en（输入，位宽度 1）：使能。当此值为 0 时，时钟触发器将被忽略。当前值继续显示在输出上。当该输入为 1 或未定义时，时钟触发器启用。➤ 南侧，用三角形表示（输入，位宽度 1）：时钟输入。当该输入值从 0 上升到 1（上升沿）时，寄存器的值将更新为 D 输入的值。➤ 南侧，标记为 0（输入，位宽度 1）：异步复位。当为 0 或未定义时，此输入无效。只要它是 1，寄存器的值就固定为 0。这是异步发生的，即与当前时钟输入值无关。只要这是 1，其他输入就不会产生任何影响。
属性	<p>选择或添加组件时，按下 Alt + 0 到 9 会更改其"数据位宽"属性。</p> <ul style="list-style-type: none">● 数据位宽：存储在寄存器中的值的位宽。● 触发：配置如何解释时钟输入。上升沿值表示寄存器应在时钟从 0 上升到 1 的瞬间更新其值。下降沿值表示应在时钟由 1 下降到 0 的瞬间更新。高电平值表示每当时钟输入为 1 时，寄存器应持续更新。低电平值表示当时钟输入为 0 时应持续更新。● 标签：与寄存器关联的标签内的文本。● 标签字体：用于呈现标签的字体。
戳工具行为	单击寄存器将键盘焦点移到寄存器（由红色矩形表示），键入十六进制数字将更改存储在寄存器中的值。
文本工具行为	允许编辑与元件关联的标签。

计数器 Counter

名称	计数器 Counter 															
行为	<p>计数器保存一个值，其值在输出 Q 上发送。每次时钟输入（用组件南边的三角形表示）根据其触发器属性触发时，计数器中的值可能会根据组件西侧的两个输入进行更新：上部输入称为加载，下部输入称为计数，其作用如下。</p> <table><tr><th><i>load</i></th><th><i>count</i></th><th>trigger action</th></tr><tr><td>0 or z</td><td>0</td><td>The counter remains unchanged.</td></tr><tr><td>0 or z</td><td>1 or z</td><td>The counter increments.</td></tr><tr><td>1</td><td>0</td><td>The counter loads the value found at the D input.</td></tr><tr><td>1</td><td>1 or z</td><td>The counter decrements.</td></tr></table> <p>可以使用最大值属性配置计数范围。当计数器达到该值时，下一个增量将计数器重置回 0；如果计数器为 0，则减量将使计数器返回最大值。</p> <p>除了输出 Q 之外，该组件还包括一个单位输出进位。当计数器处于最大值且加载和计数输入指示组件应在下一步中递增时，或当计数器处于 0 且加载和计数器输入指示在下一步骤中递减时，此值为 1。</p> <p>清除输入异步将计数器的值重置为 0（全部为零）；也就是说，只要 clr 输入为 1，该值就固定为 0，与时钟输入无关。</p>	<i>load</i>	<i>count</i>	trigger action	0 or z	0	The counter remains unchanged.	0 or z	1 or z	The counter increments.	1	0	The counter loads the value found at the D input.	1	1 or z	The counter decrements.
<i>load</i>	<i>count</i>	trigger action														
0 or z	0	The counter remains unchanged.														
0 or z	1 or z	The counter increments.														
1	0	The counter loads the value found at the D input.														
1	1 or z	The counter decrements.														
引脚	<ul style="list-style-type: none">➤ 东侧，标记为 Q（输出，位宽与数据位宽属性匹配）：输出计数器当前存储的值。➤ 东侧，下部引脚（输出，位宽度 1）：进位。当加载和计数指示递增时，只要计数器处于最大值，此输出为 1。当加载和计数指示递减时，每当计数器为 0 时，此输出为 1。在所有其他时间，此输出均为 0。➤ 西侧，上部：（输入，位宽度 1）：加载。当此值为 1 而计数输入为 0 时，计数器将加载在下一个时钟触发器的数据输入中找到的值，或者，如果计数输入恰好为 1，计数器的值将															


	<p>减小。</p> <ul style="list-style-type: none">➤ 西侧，中间，标记为 D（输入，具有匹配数据位宽属性的位）：当时钟在加载为 1 且计数为 0 时触发时，计数器的值变为此输入的值。➤ 西侧，下部，标记 ct（输入，位宽度 1）：计数。当该值为 1 或未定义时，无论何时触发时钟输入，计数器中的值都会递增，或者如果加载输入碰巧也为 1，则该值会递减。➤ 南侧，用三角形表示（输入，位宽度 1）：时钟。当触发触发器属性指定的时钟时，计数器会根据加载和计数输入的指示进行更新。➤ 南侧，标记为 0（输入，位宽度 1）：清除。当为 0 或未定义时，此输入无效。只要它是 1，计数器的值就会异步固定到 0。这是异步发生的，即与当前时钟输入值无关。只要这是 1，其他输入就没有任何影响。
属性	<p>选择或添加组件时，按下 Alt + 0 到 9 会更改其“数据位宽”属性。</p> <ul style="list-style-type: none">● 数据位宽：组件输出的值的位宽。● 最大值（Maximum Value）：最大值，此时计数器将设置进位输出。● 溢出时的操作（Action on Overflow）：计数器尝试递增超过最大值或递减超过 0 时的行为。支持四种可能的操作： 环绕（Wrap around）：下一个值为 0（如果递增，如果递减则为最大值） 保持：计数器的值保持在最大值（如果递减，则保持为 0） 继续：计数器继续递增/递减，保持“数据位宽”属性提供的位数 加载下一个值：下一个值从 D 输入加载。● 触发：配置如何解释时钟输入。值上升沿指示计数器应在时钟从 0 上升到 1 的瞬间更新其值。下降沿指示计数器应该在时钟从 1 下降到 0 的瞬间更新。● 标签：与组件关联的标签内的文本。● 标签字体：用于呈现标签的字体。
戳工具行为	<p>单击计数器将键盘聚焦到组件上（由红色矩形表示），键入十六进制数字将更改计数器中存储的值。</p>
文本工具行为	<p>允许编辑与元件关联的标签。</p>

移位寄存器 Shift Register

名称	移位寄存器 Shift Register 
行为	<p>该寄存器由几个阶段组成，其中每个时钟周期可能使每个阶段接收前一阶段的值，而一个新值加载到第一阶段。该组件还支持并行加载并存储所有阶段的值。</p> <p>清除输入异步将所有阶段重置为 0（全部为零）；也就是说，只要清除输入为 1，所有值都固定为 0，而不管时钟输入如何。</p>
引脚	<p>*标记表示仅在启用“并行加载”属性时才存在的引脚。</p> <ul style="list-style-type: none">➤ 西侧，上部（输入，位宽度 1）：移位。当 1 或断开时，所有阶段都随着时钟触发而前进；但如果为 0，则不发生任何阶段推进。如果加载输入为 1，则忽略此输入。➤ 西侧，中间（输入，位宽与数据位宽属性匹配）：数据。当推进阶段时，此输入的值将加载到第一阶段。➤ 西侧，用三角形标记的底部（输入，位宽度 1）：时钟。在触发触发器属性指定的时间时，组件可能会提前推进或加载新值。


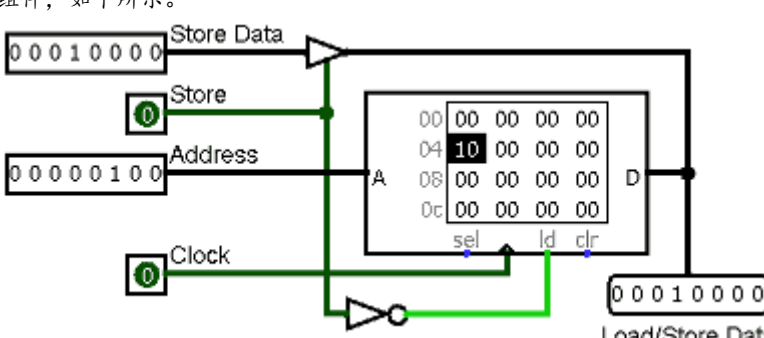
	<ul style="list-style-type: none"> ➤ *北侧，左边（输入，位宽度 1）：加载。当此值为 1 时，在其他北边管脚上的值将在下一个时钟触发时加载到所有阶段。当 0 或断开连接时，无负载发生。 ➤ *北侧，其他引脚（输入，位宽与数据位宽属性匹配）：数据。当加载输入为 1 且触发时钟时，这些值加载到所有阶段。最左边的输入对应于最前面的阶段。 ➤ 南侧，左边（输入，位宽度 1）：清除。当此值为 1 时，所有阶段异步重置为 0，并且忽略所有其他输入。 ➤ *南侧，其他管脚（输出，位宽度与数据位宽属性匹配）：输出。输出存储在每个阶段中的值，最年前面的阶段反映在最左侧的管脚上（靠近清除输入）。 ➤ 东侧（输出，位宽与数据位宽属性匹配）：输出。输出存储在最终阶段中的值。
属性	<p>选择或添加组件时，按下数字"0"到"9"会更改其"阶段数"属性，按下 Alt + 0 到 9 会更改其数据位宽属性。</p> <ul style="list-style-type: none"> ● 数据位宽：每个阶段中存储的值的位宽。 ● 阶段数：组件中包含的阶段数。 ● 并行加载：如果选择是，则组件包括上述标星号的输入和输出，以便于并行访问所有阶段的值。 ● 触发：配置如何解释时钟输入。上升沿表示寄存器应在时钟从 0 上升到 1 的瞬间更新其值。下降沿表示应在时钟由 1 下降到 0 的瞬间更新。 ● 标签：与组件关联的标签内的文本。 ● 标签字体：用于呈现标签的字体。
戳工具行为	如果 Parallel Load（并行加载）属性为 no，或者 Data Bits（数据位宽）属性大于 4，则使用戳工具按动寄存器无效。否则，单击组件将使键盘聚焦于单击的阶段（由红色矩形指示），键入十六进制数字将更改存储在该阶段中的值。
文本工具行为	允许编辑与元件关联的标签。

随机数生成器 Random

名称	随机数生成器 Random 
行为	<p>该组件迭代一个伪随机数字序列，在启用该组件的情况下，每次触发时钟时，该序列都会前进到下一个数字。从技术上讲，用于计算伪随机序列的算法是一个线性同余生成器：从种子 r0 开始，下一个数字 r1 就是</p> $r1 = (25214903917 * r0 + 11) \bmod 248$ <p>使用相同的计算方法从 r1 计算下一个值 r2，依此类推。该序列为 48 位数字；在第一次舍弃当前种子的低 12 位之后，从组件中看到的值是由其"数据位宽"属性设置的低阶位。</p> <p>除了时钟输入外，该组件还包括一个使能输入，当使能为 0 时，该输入会导致时钟输入被忽略，而重置输入会异步将组件的值重置为初始种子 r0。</p> <p>初始种子可由用户配置。如果配置为 0（默认值），则种子将基于当前时间；当重置时，组件基于新的当前时间计算新的种子。</p>
引脚	<ul style="list-style-type: none"> ➤ 东侧，标记为 Q（输出，位宽与数据位宽属性匹配）：输出组件当前存储的值。 ➤ 西侧，顶部，用三角形标记（输入，位宽度 1）：时钟。当触发触发器属性指定的时钟时，组件按其顺序前进到下一个数字。 ➤ 西侧，底部（输入，位宽度 1）：使能。当该输入断开或为 1 时，组件启用；但如果为 0，


	<p>则忽略时钟输入。</p> <p>➤ 南侧（输入，位宽度 1）：重置。当该值为 1 时，伪随机序列异步重置为初始种子。（如果种子为 0，则此新种子应与以前使用的初始种子不同。）</p>
属性	<p>选择或添加组件时，按下 Alt + 0 到 9 会更改其“数据位宽”属性。</p> <ul style="list-style-type: none">● 数据位宽：组件发出的值的位宽度。● 种子：用于伪随机序列的起始值。如果该值为 0（默认值），则起始值基于随机序列开始的时间。● 触发：配置如何解释时钟输入。值上升沿表示组件应在时钟从 0 上升到 1 的瞬间更新其值。下降沿表示组件应该在时钟从 1 下降到 0 的瞬间更新。● 标签：与组件关联的标签内的文本。● 标签字体：用于呈现标签的字体。
戳工具行为	无
文本工具行为	允许编辑与元件关联的标签。

随机访问存储器 RAM

名称	随机访问存储器 RAM 
行为	<p>RAM 组件是 Logisim 内置库中最复杂的组件，可存储多达 16777216 个值（在地址位宽属性中指定），每个值最多可包含 32 位（在数据位宽属性内指定）。电路可以在 RAM 中加载和存储值。此外，用户可以通过戳工具 Poke Tool 交互修改各个值，也可以通过 Menu Tool 修改整个内容。当前值显示在组件中。地址在显示区左侧以灰色列出。在内部，每个值都使用十六进制列出。当前所选地址的值将以反色文本显示（黑底白字）。</p> <p>RAM 组件支持三种不同的接口，具体取决于“数据接口”属性。</p> <ul style="list-style-type: none">✓ 一个同步加载/存储端口（默认）：该组件东侧包括一个单一端口，用于加载和存储数据。它执行哪种操作取决于 Id 引脚的值。为 1（或浮动）表示将数据加载到组件西侧指定的地址，而 0 表示存储端口上给定的数据。要将数据传输到组件内外，需要使用受控缓冲区组件，如下所示。  <p>The diagram illustrates a RAM component with a 4x4 grid of memory cells. The selected address is 10, which is highlighted in the grid. The RAM component has inputs for Store Data, Address, Clock, and Id. The output is connected to a Load/Store Data component.</p> <ul style="list-style-type: none">✓ 一个异步加载/存储端口：和上面一样，只是没有时钟。只要 Id 输入为 0，数据总线上找到的值就会存储到内存中。如果 Id 输入为 0 时，地址或数据发生变化，则会发生额外的一次存储。此选项旨在更贴紧许多可用随机访问存储器的接口。✓ 分立加载和存储端口：提供了两个数据端口，一个在西侧用于存储数据，另一个在东侧用于加载数据。此选项消除了处理受控缓冲区的必要性，因此更易于使用。

引脚	<ul style="list-style-type: none">➤ 西侧的 A（输入，位宽与地址位宽属性匹配）：选择电路当前正在访问内存中的哪个值。➤ 西侧的 D（输入，位宽与数据位宽属性匹配）：仅当“数据接口”属性选择“分立加载和存储端口”时，才会显示此输入。当请求存储时（时钟从 0 变为 1，而 sel 和 str 均为 1 或浮动），在该端口得到的值存储在当前选定地址的内存中。➤ 东侧的 D（输入/输出或仅输出，位宽与数据位宽属性匹配）：如果 sel 和 ld 为 1 或浮动，则 RAM 组件输出在该端口上当前选定地址处读取的值。如果有单个加载/存储端口，则每当请求存储时，都会存储从该端口读取到的值。➤ 南侧的 str（输入，位宽度 1）：存储。仅当“数据接口”属性选择“分立加载和存储端口”时，才会显示此输入。当它为 1 或浮动时，时钟脉冲将导致把在西侧得到的数据存储到内存中（前提是 sel 输入也是 1 或浮动）。➤ 南侧的 sel（输入，位宽度 1）：芯片选择。该输入根据值是 1/浮动还是 0 来启用或禁用整个 RAM 模块。该输入主要用于有多个 RAM 单元且同一时刻只有一个可以启用的情况。➤ 南侧三角形（输入，位宽度 1）：时钟输入。当“数据接口”属性位一个异步加载/存储端口时，则不存在时钟输入。在其他情况下，当 ld 为 0 时，该输入从 0 上升到 1（sel 为 1/浮动，clr 为 0），则当前所选地址的值将更改为 D 引脚的值。但是，只要时钟输入保持为 0 或 1，D 值就不会存储到存储器中。➤ 南侧的 ld（输入，位宽度 1）：加载。选择 RAM 是否应在当前地址（A）发出（引脚 D 的）值。如果 out 为 1 或未定义，则启用此输出行为；如果 out 为 0，则不会将任何值发送到 D 上，但如果存在结合复用的加载/存储端口，则会启用存储。➤ 南侧的 clr（输入，位宽度 1）：清除。当此值为 1 时，内存中的所有值都固定为 0，无论其他输入是什么。
属性	<p>选择或添加组件时，按下数字“0”到“9”会更改其地址位宽属性，按下 Alt+ 0 到 9 会更改其数据位宽属性。</p> <ul style="list-style-type: none">● 地址位宽：地址位的位宽，设为 x。RAM 中存储的值的数量为 2^x。● 数据位宽：内存中每个单独值的位宽。● 数据接口：配置三个接口中的哪一个用于将数据传入和传出。
戳工具行为	请参见《用户指南》中的 poking memory。
文本工具行为	无
菜单工具行为	请参见《用户手册》中的弹出菜单和文件。









只读存储器 ROM

名称	只读存储器 ROM 
行为	<p>ROM 组件最多可存储 16777216 个值（在地址位宽属性中指定），每个值最多可包含 32 位（在数据位宽属性内指定）。电路可以访问 ROM 中的值，但不能更改它们。用户可以通过 Poke Tool 交互修改单个值，也可以通过 Menu Tool 修改整个内容。</p> <p>与 RAM 组件不同，ROM 组件的当前内容属于组件的属性。因此，如果包含 ROM 组件的电路使用两次，则两个 ROM 组件内保持相同的值。</p> <p>ROM 当前值显示在组件中。显示的地址在显示区左侧以灰色列出。在内部，每个值都使用十六</p>


	进制列出。当前所选地址的值将以反色文本显示（黑底白字）。
引脚	<ul style="list-style-type: none">➤ 西侧的 A（输入，位宽与地址位宽属性匹配）：选择电路当前正在访问的值。➤ 东侧的 D（输入/输出，位宽与数据位宽属性匹配）：如果 sel 为 1 或浮动，则在 D 引脚的当前选定地址处输出值。如果 sel 为 0，则 D 将浮动。➤ 南侧的 sel（输入，位宽度 1）：如果只有一个 ROM 模块，请忽略此输入。如果您有多个并行的 ROM 模块，您可以使用该输入来启用或禁用整个 ROM 模块（基于值是 1 还是 0）。换句话说，当该值为 0 时，D 输出上不会发出任何值。
属性	<p>选择或添加组件时，按下数字“0”到“9”会更改其地址位宽属性，按下 Alt + 0 到 9 会更改其数据位宽属性。</p> <ul style="list-style-type: none">● 地址位宽：地址位的位宽，设为 x。ROM 中存储的值的数量为 2^x。● 数据位宽：内存中每个单独值的位宽。● 内容（Contents）：存储内存的具体内容。
戳工具行为	请参见《用户指南》中的 poking memory。
文本工具行为	无
菜单工具行为	请参见《用户手册》中的弹出菜单和文件。

第六节：输入输出库 Input/Output library

输入/输出库包括与电子设备中的典型组件相对应的组件，用于与用户交互。


-  [Button](#)
-  [Joystick](#)
-  [Keyboard](#)
-  [LED](#)
-  [7-Segment Display](#)
-  [Hex Digit Display](#)
-  [LED Matrix](#)
-  [TTY](#)

按钮 Button


名称	按钮 Button 
行为	正常输出 0；但当用户使用 Poke Tool 按下按钮时，输出为 1。
引脚	一个按钮只有一个引脚，一个 1 位输出。
属性	<p>选择或添加零部件时，按下方向键会更改其“方向”属性。</p> <ul style="list-style-type: none">● 方向：输出引脚相对于组件的位置。● 颜色：用于显示按钮的颜色。● 标签：与组件关联的标签内的文本。● 标签位置：标签相对于组件的位置。● 标签字体：用于呈现标签的字体。● 标签颜色：用于绘制标签的颜色。

戳工具行为	按下鼠标按钮时，组件的输出将为 1。松开鼠标按钮后，输出将恢复为 0。
文本工具行为	允许编辑与元件关联的标签。

摇杆 Joystick


名称	摇杆 Joystick 
行为	用户可以在正方形区域内拖动圆形红色旋钮，输出会更新以指示旋钮的当前 x 和 y 坐标。这是为了模仿古典街机游戏时代的操纵杆。
引脚	<ul style="list-style-type: none">➤ 西侧北端（输出，位宽与位宽属性匹配）：表示旋钮的 x 坐标，被解释为一个无符号整数，其值永远不会为 0。因此，值 1 表示最左边，位宽的最大值表示最右边。当旋钮处于静止状态（在中心）时，该值的位模式为 10...00。➤ 西侧南端（输出，位宽与数据位宽属性匹配）：表示旋钮的 y 坐标，其值范围与 x 坐标相同。当旋钮被拉到顶部时，该输出值为 1，当旋钮被拉到底部时，输出为所选位宽的最大值。
属性	选择或添加组件时，按下 Alt + 2 到 5 会更改其“位宽”属性。 <ul style="list-style-type: none">● 位宽：用于指示每个旋钮坐标共有多少位。● 颜色：屏幕上绘制的旋钮颜色。
戳工具行为	在操纵杆区域内按下鼠标按钮，将旋钮移动到该位置并更新输出。拖动鼠标继续移动旋钮并更新输出，使旋钮保持在操纵杆区域内。松开鼠标按钮可将旋钮恢复至其静止位置。
文本工具行为	无

键盘 Keyboard


名称	键盘 Keyboard 
行为	<p>该组件允许电路读取从键盘键入的键——只要键可以用 7 位 ASCII 码表示。使用戳工具 poke tool 单击组件后，用户可以键入字符，这些字符累积在缓冲区中。在任何时候，缓冲区中最左侧字符的 ASCII 值都会发送到最右侧的输出。当时钟输入被触发时，最左边的字符从缓冲区中消失，新的最左边字符被发送到最右边的输出。</p> <p>缓冲区支持的字符包括所有可打印的 ASCII 字符，以及空格、换行符、退格符和 control-L。此外，左方向键和右方向键在缓冲区内移动光标，删除键删除光标右侧的字符（如果有）。</p> <p>该组件是异步的，即当缓冲区为空并且用户键入字符时，该字符会立即作为输出发送，而无需等待时钟脉冲。</p>
引脚	<ul style="list-style-type: none">➤ 西侧，用三角形标记（输入，位宽度 1）：时钟。当读取使能引脚不为 0 时且时钟触发时，将删除缓冲区中最左侧的字符，并更新输出以反映缓冲区的新状态。➤ 南侧，最左边引脚（输入，位宽度 1）：读取使能——当为 1（或浮动或错误）时，时钟边沿将消耗缓冲区中最左边的字符。当读取使能为 0 时，时钟输入被忽略。➤ 南侧，左起第二个引脚（输入，位宽度 1）：清除。当为 1 时，缓冲区被清空，不接受其他

	<p>字符。</p> <ul style="list-style-type: none">➤ 南侧，右起第二个引脚（输出，位宽度 1）：可用。当缓冲区至少包含一个字符时为 1，当缓冲区为空时为 0。➤ 南侧，最右侧引脚（输出，位宽度 7）：数据。缓冲区中最左侧字符的 7 位 ASCII 代码，如果缓冲区为空，则为 0。
属性	<ul style="list-style-type: none">● 缓冲区长度（Buffer Length）：缓冲区一次可以容纳的字符数。● 触发：如果该值为上升沿，则当时钟输入从 0 变为 1 时，最左边的字符被消耗（当读取使能不为 0 时）。如果它是下降沿，那么当时钟输入从 1 变为 0 时就会发生这种情况。
戳工具行为	使用戳工具单击组件可使键盘聚焦组件，并显示一个垂直光标。
文本工具行为	无

发光二极管 LED


名称	发光二极管 LED 
行为	根据输入是 1 还是 0，通过给 LED 上色（由其“颜色”属性指定）来显示输入值。
引脚	LED 只有一个引脚，一个 1 位输入，用于确定是显示彩色 LED（当输入为 1 时）还是暗 LED（当输出为其他输入时）。
属性	<p>选择或添加零部件时，按下方向键会更改其“方向”属性。</p> <ul style="list-style-type: none">● 方向：输入引脚相对于组件的位置。● 颜色：输入值为 1 时显示的颜色。● 高电平时激活？（Active on High?）：如果是，则当输入为 1 时 LED 为彩色。如果不是，则当输入为 0 时 LED 为有色。● 标签：与组件关联的标签内的文本。● 标签位置：标签相对于组件的位置。● 标签字体：用于呈现标签的字体。● 标签颜色：用于绘制标签的颜色。
戳工具行为	无
文本工具行为	允许编辑与元件关联的标签。

七段数码管 7-Segment Display


名称	七段数码管 7-Segment Display 
行为	显示其八个一位输入的值。根据输入，线段可以是彩色的，也可以是浅灰色的。
引脚	<ul style="list-style-type: none">➤ 北侧，左起第一个（输入，位宽度 1）：控制中间水平段。➤ 北侧，左起第二个（输入，位宽度 1）：控制左侧的上部垂直线段。➤ 北侧，左数第三个（输入，位宽度 1）：控制上水平段。

	<ul style="list-style-type: none">➤ 北侧，左起第四个（输入，位宽度 1）：控制右侧的上部垂直线段。➤ 南侧，左起第一个（输入，位宽度 1）：控制左侧的下部垂直线段。➤ 南边，左起第二个（输入，位宽度 1）：控制底部水平段。➤ 南边，左数第三个（输入，位宽度 1）：控制右侧的下部垂直线段。➤ 南边，左起第四个（输入，位宽度 1）：控制小数点。
属性	<ul style="list-style-type: none">● 启用颜色：打开显示段和小数点时用来绘制它们的颜色。● 关闭颜色：关闭显示段和小数点时用来绘制它们的颜色。● 背景颜色：用于绘制显示背景的颜色（默认情况下为透明）。● 高电平时激活？：如果是，则当相应的输入为 1 时，线段亮起。如果不是，则当对应的输入为 0 时，线段亮起。
戳工具行为	无
文本工具行为	无

十六进制数码管 Hex Digit Display


名称	十六进制数码管 Hex Digit Display 
行为	使用七段数码管，显示与四位输入对应的十六进制数字。如果任何一位输入不是 0/1（浮动或错误），则显示屏显示破折号（“-”）。单独的一位输入控制小数点的显示。
引脚	<ul style="list-style-type: none">➤ 南侧，左起第一个（输入，位宽度 4）：此输入被解释为无符号四位数字，并显示相应的十六进制数字。如果任何位为浮动或错误，则显示破折号（“-”）。➤ 南侧，左起第二个（输入，位宽度 1）：控制小数点。如果不连接，小数点保持关闭。
属性	<ul style="list-style-type: none">● 启用颜色：打开显示段和小数点时用来绘制它们的颜色。● 关闭颜色：关闭显示段和小数点时用来绘制它们的颜色。● 背景颜色：用于绘制显示背景的颜色（默认情况下为透明）。
戳工具行为	无
文本工具行为	无

LED 矩阵 LED Matrix

名称	LED 矩阵 LED Matrix 
行为	显示由像素组成的小网格，其值由当前输入确定。网格最多可以有 32 行 32 列。
引脚	<p>组件的界面因“输入格式”属性的值而异。有三种格式：</p> <ul style="list-style-type: none">➤ 柱：输入沿组件的南边排列，矩阵的每列有一个多位输入。每个输入的位数与矩阵中的行数一样多，最低位对应列中最南端的像素。1 表示点亮相应的像素，而 0 表示保持像素暗淡。如果列的任何位是浮动值或错误值，则列中的所有像素都会亮起。➤ 排：输入沿组件的西边排列，矩阵的每一行有一个多位输入。每个输入的位数与矩阵中的

	<p>列一样多，最低位对应行中最右边的像素。与柱格式一样，1 表示点亮相应的像素，0 表示保持像素暗淡。如果一行的任何位是浮动值或错误值，那么该行中的所有像素都会亮起。</p> <p>➤ 选择行/列：组件的西边有两个输入端。上面的多位输入的位数与矩阵中的列一样多，最低位对应最右边的列。下面的多位输入的位数与矩阵中的行数一样多，低位对应于底行。如果任一输入中的任何位是浮动值或错误值，矩阵中的所有像素都会亮起。通常情况下，如果上部输入中的相应列位为 1，下部输入中的对应行位为 1 时，特定行-列位置的像素会点亮。例如，对于 5x7 矩阵，如果第一个输入为 01010，第二个输入为 0111010，则第二和第四列会点亮第二、第三、第四和第六行；结果似乎是一对感叹号。（这种输入格式似乎不直观，但 LED 矩阵正是通过这种接口在商业上销售的。例如，Lite On 销售此类组件。）</p>
属性	<ul style="list-style-type: none">● 输入格式（创建组件后只读，不可修改）：如上所述，选择管脚与像素的对应方式。● 矩阵列：选择矩阵中的列数，范围从 1 到 32。● 矩阵行：选择矩阵中的行数，范围从 1 到 32。● 启用颜色：选择像素点亮时的颜色。● 关闭颜色：选择像素暗淡时的颜色。● 灯光持久性（Light Persistence）：当该值不是 0 时，在组件的输入指示像素应该变暗后，点亮的像素在给定的时钟周期数内保持点亮。● 点形状（Dot Shape）：正方形选项意味着每个像素都绘制为 10x10 正方形，在像素之间填充组件，不留任何间隙。圆选项意味着每个像素被绘制为直径为 8 的圆，每个圆之间有间隙。圆形选项更难理解，但它更接近现成的 LED 矩阵组件。
戳工具行为	无
文本工具行为	无

控制终端 TTY

名称	控制终端 TTY 
行为	<p>该组件实现了一个非常简单的哑终端。它接收 ASCII 码序列并显示每个可打印字符。当当前行变满时，光标会移动到下一行，如果光标已经位于最下面一行，则可能会向上滚动所有当前行。唯一支持的控制方式是：退格（ASCII 8），它删除最后一行中的最后一个字符，除非最后一行已经为空；换行符（ASCII 10），将光标移动到下一行的开头，必要时滚动；和换页（ASCII 12，键入 control-L），清除屏幕。</p>
引脚	<ul style="list-style-type: none">➤ 西侧，上方（输入，位宽为 7）：数据。这是要输入终端的下一个字符的 ASCII 值。➤ 西侧，用三角形标记（输入，位宽度 1）：时钟。当写入使能引脚不为 0 时触发时，终端处理数据输入上方的当前 ASCII 值。➤ 南侧，最左侧引脚（输入，位宽度 1）：写入使能。当为 1（或浮动或错误）时，时钟边沿将导致处理数据输入中的新字符。当写入使能为 0 时，时钟和数据输入被忽略。➤ 南侧，左起第二个引脚（输入，位宽度 1）：清除。当为 1 时，终端将清除所有数据，并忽略所有其他输入。
属性	<ul style="list-style-type: none">● 排：终端中显示的行数。● 项目数（Columns）：每行终端中显示的最大字符数。


	<ul style="list-style-type: none">● 触发：如果该值为上升沿，则当时钟输入从 0 变为 1 时，将处理数据输入（当写入使能不为 0，清除输入不启用时）。如果它是下降沿，那么当时钟输入从 1 变为 0 时就会发生这种情况。● 颜色：用于绘制终端中显示的文本的颜色。● 背景颜色：用于绘制终端背景的颜色。
戳工具行为	无
文本工具行为	无

第七节：基础库 Base library


基本库包括通用工具。

-  [Poke Tool](#)
-  [Edit Tool](#)
-  [Select Tool](#)
-  [Wiring Tool](#)
-  [Text Tool](#)
-  [Menu Tool](#)
-  [Label](#)

戳工具 Poke Tool

名称	戳工具 Poke Tool 						
行为	<p>戳工具用于操作与组件关联的当前值。戳工具的精确行为取决于单击的组件；此行为记录在每个单独组件的“Poke Tool behavior”部分中。以下组件都支持戳工具。</p> <table><tr><td>Base library</td><td>Pin Clock</td></tr><tr><td>Memory library</td><td>D/T/J-K/S-R Flip-Flop Register Counter Shift Register RAM ROM</td></tr><tr><td>Input/Output library</td><td>Button Joystick Keyboard</td></tr></table> <p>此外，使用戳工具单击导线段会显示导线当前携带的值，如线工具 Wiring Tool 页面中所述。</p>	Base library	Pin Clock	Memory library	D/T/J-K/S-R Flip-Flop Register Counter Shift Register RAM ROM	Input/Output library	Button Joystick Keyboard
Base library	Pin Clock						
Memory library	D/T/J-K/S-R Flip-Flop Register Counter Shift Register RAM ROM						
Input/Output library	Button Joystick Keyboard						
属性	没有，但是单击支持 Poke Tool 的组件将显示该组件的属性。						

编辑/选择工具 Edit/Select Tool

名称	编辑/选择工具 Edit Tool 
行为	“编辑”工具允许用户重新排列现有组件并添加导线。该工具的具体功能取决于用户在画布上按下鼠标的位置。

- ✓ 当鼠标位于现有组件的布线点上方时，或者如果鼠标位于当前导线上方，则“编辑工具”（Edit Tool）将在鼠标位置周围显示一个小的绿色圆圈。按下此处的按钮，开始添加新导线。但是，如果用户在释放按钮之前没有拖动鼠标足够长的距离来启动连线，则按下会被视为鼠标单击，因此连线只会被添加到当前选择中。

添加的导线的位宽度是从其连接的组件推断出来的。如果未连接到任何组件，则导线将绘制为灰色，表示其位宽度未知；如果导线帮助连接的位置上的组件在位宽度上不一致，则导线将绘制为橙色以指示冲突，并且导线实际上将无法携带任何值，直到用户解决冲突。Logisim 中的所有导线都是水平或垂直的。


导线是非定向的；也就是说，它们将值从一个端点传递到另一个端点。实际上，导线可以同时两个方向上传输值：在下面的示例中，一个 bit 从左上方的输入端流过中心导线，然后再穿过中心导线绕回来，然后再向前穿过中心导线，最后到达右下方的输出端。



- ✓ 拖动鼠标一次可以创建多个导线段。准确的描述这个过程有些困难；但它在实践中非常直观地工作：如果使用线工具 Wiring Tool 请求特定的导线段，该段将在其碰到现有元件的接点或现有导线段的端点的任何位置被分割。此外，如果任何新导线段的端点触及现有导线的中间某处，则该导线本身将分割为多个段。
- ✓ 也可以通过在线束段的端点处开始拖动，然后在线束段上向后绘制，来缩短或删除现有线束段。在拖动过程中，通过在将要删除的导线部分上绘制一条白线来指示缩短。
- ✓ 某些组件绘制可以连接导线的短线，例如或门和受控缓冲区。Logisim 会纠正创建稍微超出存根末端的导线的尝试。
- ✓ 但是，如果用户在导线中间的某个点按 Alt 键，则绿色圆圈将消失。按住鼠标选择导线，然后拖动鼠标移动导线。
- ✓ 当鼠标按钮位于当前选定组件内时，按下鼠标按钮将开始拖动移动选定的所有元素。默认情况下，Logisim 将计算添加新导线的方法，以便在移动过程中不会丢失现有连接。（有时会删除或缩短现有导线。）如果您正在执行不希望进行这些更改的移动，可以在移动过程中按 shift 键。如果要完全禁用此行为，请转到“项目”>“选项”，选择“画布”选项卡，然后取消选中“移动时保持连接”框；在这种情况下，仅当按下 shift 键时计算连接。
- ✓ 拖动选择可能会导致导线出现意外行为：如果拖动的选择包括其他一些导线顶部的某些导线，则会合并所有导线，并将合并的导线放置到选择中。因此，如果再次拖动选择，先前位于该位置的导线将不会落在后面。此行为对于保持 Logisim 中导线的直观行为是必要的，因为导线从不重叠。而且它通常不会构成一个重要的问题：Logisim 会在拖放过程中画出完整的选择，在您确定它位于正确位置之前，您不应该将其拖放。
- ✓ 在未选定的组件内（但不在组件的某个接线点处）按鼠标将从当前选择中删除所有组件，并选择包含单击位置的组件。
- ✓ 按住 Shift 键并在组件中单击鼠标，可以切换该组件在选择中的存在状态。如果多个组件包含同一位置，则将切换所有组件的存在。
- ✓ 从不包含在任何组件中的位置开始拖动鼠标，将从当前选择中删除所有组件，并启动矩形选择。矩形包含的所有组件都将被放置到所选内容中。
- ✓ 按住 Shift 键并从不包含在任何组件中的位置开始拖动鼠标，将启动矩形选择。将切换矩形包含的所有组件在选择中的存在。
- ✓ 但是，如果在不包含在任何组件中的位置按下 Alt 键，则会启动新导线的添加。在这种情况下，会画一个小的绿色圆圈来表示这一点。

	<p>在选择选中所需项目后，当然可以通过编辑菜单剪切/复制/粘贴/删除/复制所有项目。某些按键具有编辑工具的效果。</p> <ul style="list-style-type: none">✓ 方向键更改选择中具有此属性的所有组件的“方向”属性。✓ Delete 和 Backspace 键将从电路中删除所选内容中的所有内容。✓ Insert 和 MenuKey-D 键将创建当前选定组件的副本。 <p>Logisim 在复制选择或将剪贴板粘贴到电路中时的行为有点奇怪：它不会立即将元件放置到回路中；相反，先被放置的将是一组“重影”，一旦它们被拖到另一个位置或从选择中删除，它们就会被放到电路中。（这种特殊行为是必要的，因为粘贴会将所选导线立即合并到当前回路中，如果用户想将粘贴的元件移动到其他地方，则以前所在回路中的导线将与粘贴的剪贴板一起拖动。）</p>
属性	<p>没有，但是选择组件将显示其属性。选择多个组件后，将显示所有组件共享的属性，如果它们具有各不同的值，则为空。（如果选择中有任何非导线的组件，则忽略选中的导线。）属性值的更改会影响所有选定组件。</p>

线工具 Wiring Tool


名称	线工具 Wiring Tool 
行为	基本与选择/编辑工具中有关导线描述一致。
属性	<p>布线工具本身没有属性，但它创建的导线有属性。</p> <ul style="list-style-type: none">➤ 方向：指示导线是水平还是垂直。无法更改此属性的值。➤ 长度：指示导线的长度为多少像素。无法更改此属性的值。
戳工具行为	<p>使用戳工具单击现有导线段时，Logisim 会显示通过该导线的当前值。该行为对于多位导线特别有用，因为其黑色无法提供有关导线承载的值的视觉反馈。</p> <p>对于多位值，可以使用 Logisim 首选项对话框的布局窗格精确配置值的显示方式（例如二进制、十进制或十六进制）。</p>

文本工具 Text Tool

名称	文本工具 Text Tool 						
行为	<p>文本工具允许您创建和编辑与组件关联的标签。哪些组件支持标签，请参见其文档的“文本工具行为”部分。从当前版本开始，内置库中的以下组件支持标签。</p> <table><tr><td>Base library</td><td>Pin Clock Label Probe</td></tr><tr><td>Memory library</td><td>D/T/JK/SR Flip-Flop Register Counter Shift Register Random</td></tr><tr><td>Input/Output library</td><td>Button LED</td></tr></table> <p>对于可以使用标签但当前未指定标签的组件，可以单击组件中的任意位置以添加标签。如果已经有标签，则需要在标签内单击。如果单击当前没有要编辑的标签的点，Logisim 将启动添加新标签组件。</p>	Base library	Pin Clock Label Probe	Memory library	D/T/JK/SR Flip-Flop Register Counter Shift Register Random	Input/Output library	Button LED
Base library	Pin Clock Label Probe						
Memory library	D/T/JK/SR Flip-Flop Register Counter Shift Register Random						
Input/Output library	Button LED						

	在当前版本的 Logisim 中，文本编辑功能仍然相当原始。无法在标签内选择文本区域。无法将换行符插入标签。
属性	该工具的属性与标签组件的属性相同。在编辑现有组件上的标签时，这些属性无效，但它们会被赋予使用文本工具创建的任何标签。 单击支持文本工具的组件将显示该组件的属性。

菜单工具 Menu Tool

名称	菜单工具 Menu Tool 
行为	<p>菜单工具允许用户为已经存在的组件调出一个弹出菜单。默认情况下，右键单击或控制单击组件将弹出此菜单；但是，“项目”选项的“鼠标”选项卡允许用户配置鼠标按钮以不同的方式工作。大多数组件的弹出菜单有两个项目。</p> <ul style="list-style-type: none">✓ 删除：从回路中删除元件。✓ 显示属性：将组件的属性放置到窗口的属性表中，以便可以查看和更改属性值。 <p>但是，对于某些组件，菜单中有其他项目。子电路（即使用一个电路作为另一个电路中的“黑匣子”的实例）就是这样的例子：除了上述两个项目外，弹出菜单还包括另一个项目。</p> <ul style="list-style-type: none">✓ 视图 XXX：将正在查看和编辑的电路布局更改为子电路的布局。布局中看到的值将与主电路的值属于同一层次结构的一部分。（请参阅《用户指南》的“调试子电路”部分。） <p>其他组件也可以扩展弹出菜单。在当前版本 Logisim 的内置库中，只有 RAM 和 ROM 是这样的组件。</p>
属性	无

标签 Label

名称	标签 Label
行为	<p>这是一个简单的文本标签，可以放置在回路中的任何位置。它不会以任何方式与通过电路的值交互，除了在绘制电路时它是可见的。</p> <p>与当前内置库中的所有其他组件相比，标签组件可以放在画布上的任何位置；它们不会与网格对齐。</p>
引脚	无
属性	<p>文本：标签中显示的文本。该值可以在属性表中编辑，也可以使用文本工具在画布上编辑。</p> <p>字体：绘制标签时使用的字体。</p> <p>水平对齐方式：相对于标签官方位置（在创建标签时单击鼠标的位置）的文本水平定位技术。“左”表示应绘制文本，使其左边框位于该位置；“右”是指应绘制文本，使其右边框位于该位置；而“中心”意味着应该绘制文本，使其中心（水平）位于该位置。</p> <p>垂直对齐方式：文本相对于标签官方位置（在创建标签时单击鼠标的位置）的垂直定位技术。“基准（Base）”是指基线应与位置相交；“顶部”表示文本顶部应与位置相交；“底部”表示文本底部应与位置相交；“居中”意味着文本应在该位置居中（垂直）。</p> <p>文本的顶部和底部是根据字体的标准上升和下降值计算的；因此，即使实际文本不包含高字母（例如 b）或下伸字母（例如 g），出于垂直定位的目的，也假定它包含这样的字母。</p>

戳工具 行为	无
文本工 具行为	允许编辑标签中显示的文本。