Logisim 自动化方法概览

本节内容属于选学内容，且方法并不完善，请各位同学适当参考！

自动化方法概览

在此之前我们使用 Logisim 搭建我们使用的数字电路时，都是在图形化界面（GUI）中进行操作，但在面临较为复杂，还有诸多重复性部件的电路时，如通用寄存器堆（GPR）等，如果全部使用手工搭建，会非常耗时。但如果我们考察电路的实际储存形式，可以发现它只是 XML 代码，因此可以使用代码自动生成的方法简化操作，达到提高效率的目的，接下来我们就对方法本身进行介绍，并针对某个具体问题进行讲解。

circ 文件与标签

Logisim 使用的文件是 .circ 文件，它是使用扩展性标记语言编写的文件，Logisim 通过这种文件来存储电路。这是一种描述性文件，和网页用的语言 HTML 类似，可以通过直接修改文件来更改电路图。作为一种 XML 文件，它主要是被用于**传输数据**，而非显示数据，这是与 HTML 的不同之处，因此它的标签均是可以自定义的。只需满足如 `<foo>bar</foo>` 式的格式即可，在 Logisim 中同样也有 Logisim 自行规定的一些标签，通过标签的类型其中比较常见的有：

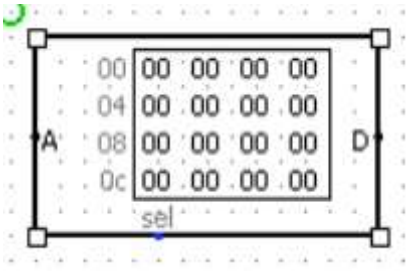
- `<circuit>` 是电路或子电路的标签，用于标记整个电路
- `<wire>` 标签，用于连线，通过 `x-y` 属性定位，需要自己尝试。
- `<comp>` 标签，拥有 `loc` 和 `name` 属性，用于调用库元件

我们的自动化方法其实是一种半自动方法，需要我们去观察使用 `GUI` 工具搭建出来的电路，再在这个基础上进行代码生成，其实是一种**聪明的复制粘贴。**

以内存矩阵为例

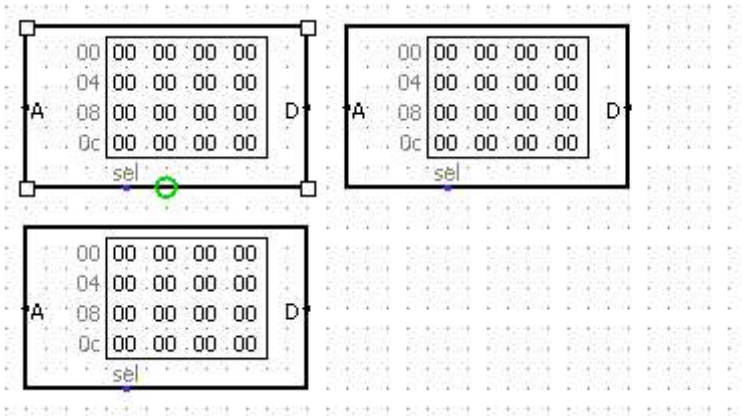
下面我们来考虑这么一个问题：

假设我们现在有 8 位字长，存储容量为 256B（8 位地址）的 ROM 芯片，我们需要搭建合理的电路，将其扩展成，32 位字长，存储容量为 16KB 的 ROM（12 位地址）。ROM 芯片如图：



从理论上分析，字长扩展了 4 倍，地址扩展了 4 位，因此共需 4 * 16 = 64 块原芯片，预计是要摆出一个 16 行 4 列的芯片矩阵。看，面对如此大量的芯片，传统地在 Logisim 中“手撸”一个电路就显得十分繁琐了。因此我们需要采用自动化的方法。

首先我们需使用 GUI 工具拖放元件，这一步其实是非常重要的，通过合理的尝试，我们可以找到显示效果最漂亮的摆放方式，为之后的编码提供位置信息。



之后我们需要观察产生的 .circ 文件，文件中我们可以发现，`<comp>` 标签表示对库元件的调用，用 `<a>` 标签表示相关的属性，`loc` 属性表示位置.每个 `<comp>` 标签就是一个一个 ROM 芯片。我们自动化方法的核心其实就是构造一个可以被重复摆放的**最小单元**，而且这个最小单元往往通过 Tunnel 连带着输出一起。在本例中即是这个 `<comp>` 标签

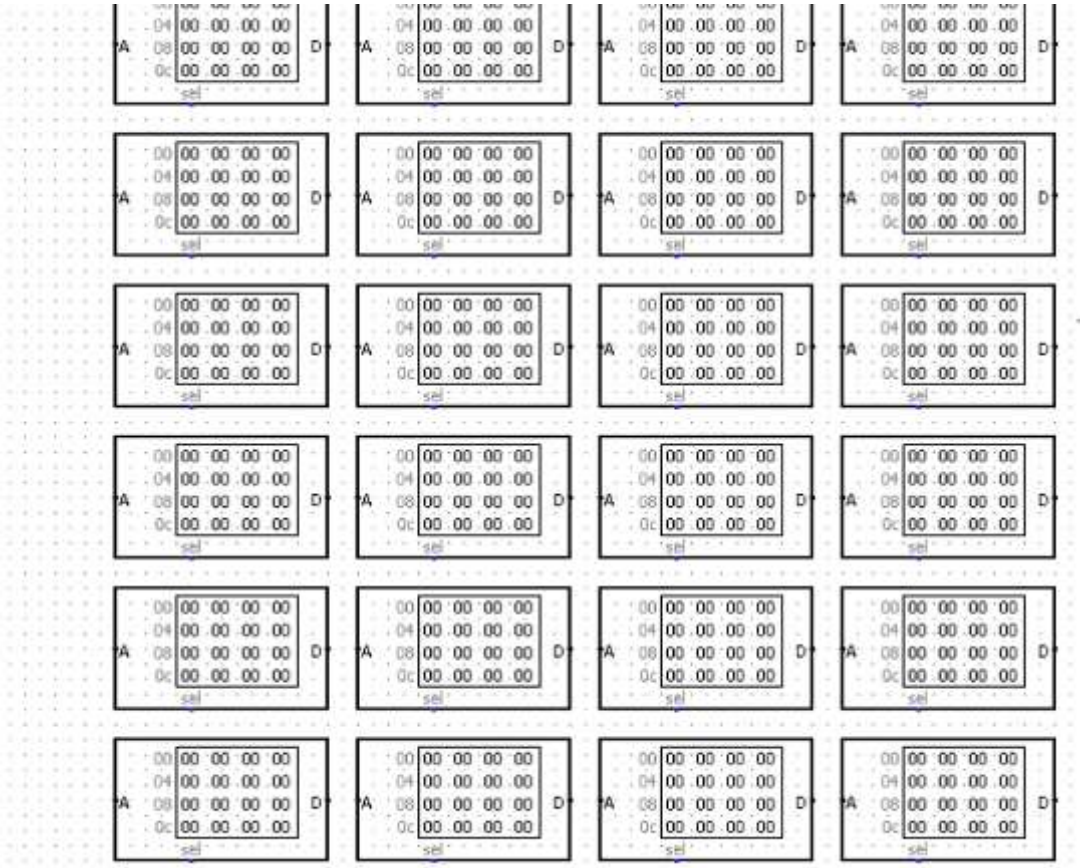
```
<comp lib="4" loc="(420,290)" name="ROM">
      <a name="contents">addr/data: 8 8
0
</a>
    </comp>
    <comp lib="4" loc="(260,390)" name="ROM">
      <a name="contents">addr/data: 8 8
0
</a>
    </comp>
    <comp lib="4" loc="(260,290)" name="ROM">
      <a name="contents">addr/data: 8 8
0
</a>
```

接着我们需要编写脚本去批量生成按照规律摆放的 ROM 芯片，在此我们使用的较为简单的 Python 语言去编写脚本。需要知道的是，整个自动化方法的核心就是把摆放电路这件**机械重复**

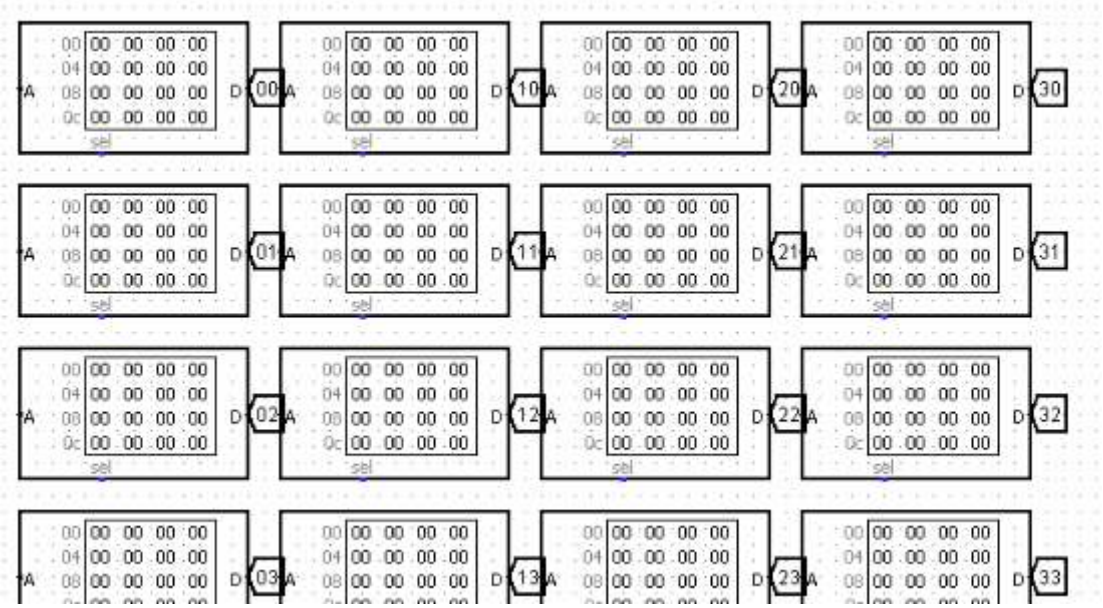
的事，交由计算机去完成。我们只需要根据前文所确定的**最小单元**，与摆放的**位置信息**，就能够编写相应的代码来自动生成所要的部分电路 XML。

```
start_x = 260
start_y = 290
step_x = 160
step_y = 100

template = """
    </comp>
        <comp lib="4" loc="(%d,%d)" name="ROM">
        <a name="contents">addr/data: 8 8
0
</a>
    </comp>
"""

result = ""
for i in range(0,4):
    for j in range(0,16):
        result += template % (start_x + step_x * i,start_y + step_y * j)

print(result)
```

观察所编写的代码，其实只是将最小单位的位置信息参数化，进行重复输出而已，编写起来应该比较简单。之后我们只需将生成的 XML 黏贴到源文件中 main 下合适的部分即可。最终效果如下：



可以注意到的是，此时我们还没有解决连线的大问题，因此我们就要使用 Tunnel 简化布线，Tunnel 元件直接不用 wire 连接在部件上组成**最小单位**，再进行自动化方法即可。效果如图，可以发现 Tunnel 的名字同样也参数化了，这样就可以通过有规律的 Tunnel 信号进行操作。



总结

综上，我们可以发现自动化方法的一般步骤就是**尝试——发现最小单元——编写生成代码——使用 XML 代码**，适用于减少重复性操作的场景，当然这种方法并不完善有许多值得改进与发展的地方，希望各位加油!