实验 2：组合逻辑电路设计

一、实验目的

1. 掌握组合逻辑电路的设计方法和步骤，实现译码器、编码器等基本组合逻辑电路。
2. 掌握串行加法器设计方法，理解减法和比较运算的实现方法。
3. 掌握汉明码校验电路的设计方法。

二、实验环境

Logisim 2.16

三、实验内容

1. 译码器实验

根据如图 2.1 所示的 3-8 译码器芯片 74X138 的电路原理图，设计一个由反相逻辑门电路构成的 3-8 译码器，并对电路进行仿真测试，以验证电路的功能。

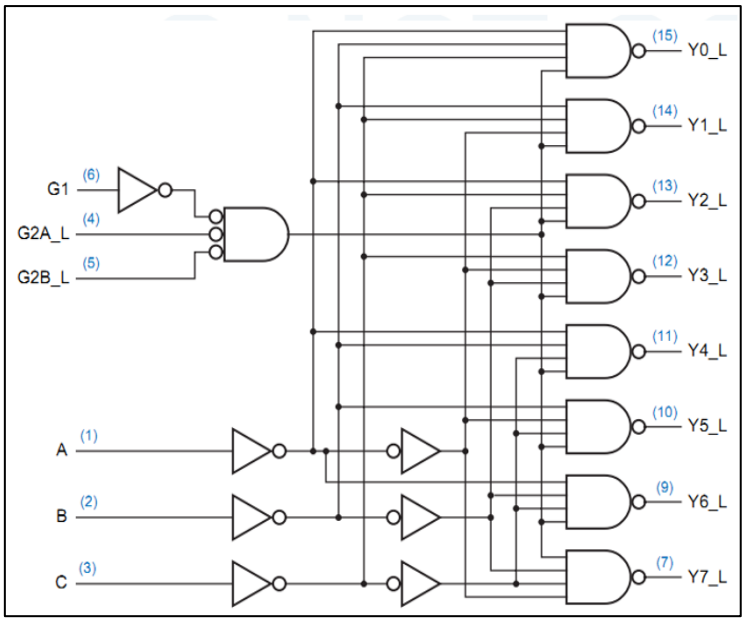

图 2.1　3-8 译码器 74X138 原理图

实验步骤如下：

1） 根据图 2.1 所示原理图添加逻辑门。选择 8 个 4 输入与非门、7 个非门、1 个与门、7 个输入引脚、8 个输出引脚，并将上述元件布局到 Logisim 的工作区中适当位置。可通过以下方式设置输入端口数：以与非门为例，选择某个与非门，在属性窗口的输入端口数输入框中设置数字 4。

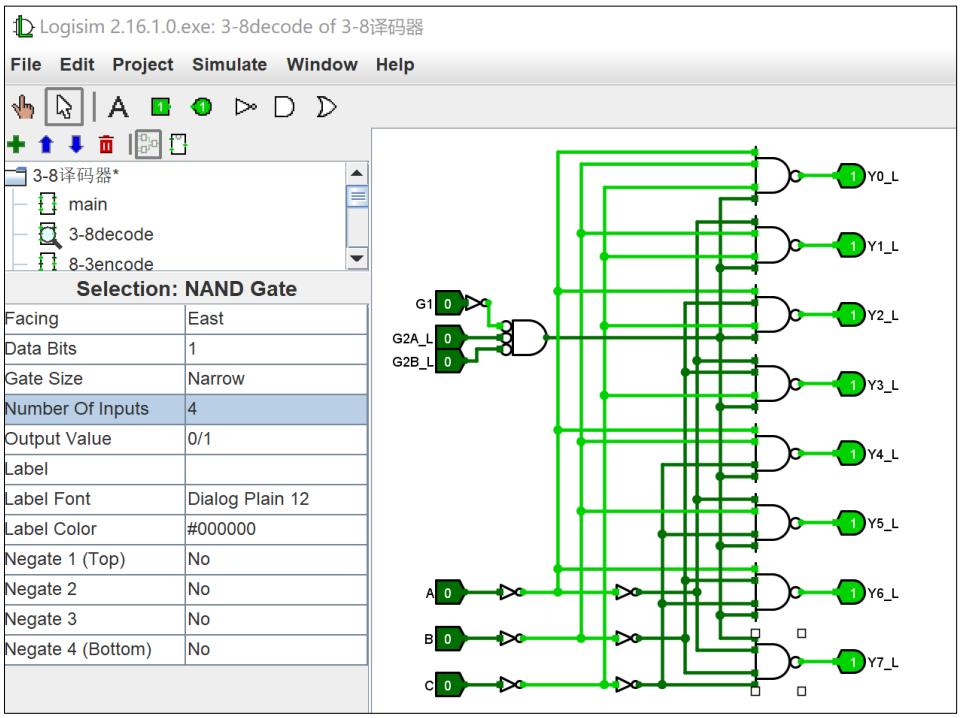

图 2.2 与非门属性窗口

2） 添加连线。将输入引脚、逻辑门的输入端、输出端、输出引脚等通过连接线相连。

3） 添加标识符。选中输入/输出引脚，在属性表中添加引脚标识符；选中逻辑门，在属性表中添加逻辑门标识符；点击快捷工具栏中的文本工具，在电路空白处添加电路的描述文字。标识符、注释文字的字体、大小、颜色和位置等均可在属性表中修改，如图 2.2 中右侧所示。

4） 仿真测试。进入仿真状态，改变输入引脚赋值，记录输出引脚值，填写表 2.1 所示的译码器功能表以验证实验结果。保存电路，文件名为 lab2，可修改导航区中的 mian 子电路为：3-8 译码器）。

表 12.1 74X138 功能表

输 入						输 出							
G1	G2A_L	G2B_L	C	B	A	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
X	1	X	X	X	X								
X	X	1	X	X	X								
0	X	X	X	X	X								
1	0	0	0	0	0								
1	0	0	0	0	1								
1	0	0	0	1	0								
1	0	0	0	1	1								
1	0	0	1	0	0								
1	0	0	1	0	1								
1	0	0	1	1	0								
1	0	0	1	1	1								

2. 编码器实验

根据如图 2.3 所示的 8-3 优先级编码器原理图，设计一个由逻辑门电路构成的 8-3 优先级编码器，并将编码器输出连接到一个十六进制数码管，通过数码管的输出显示来验证和测试电路。

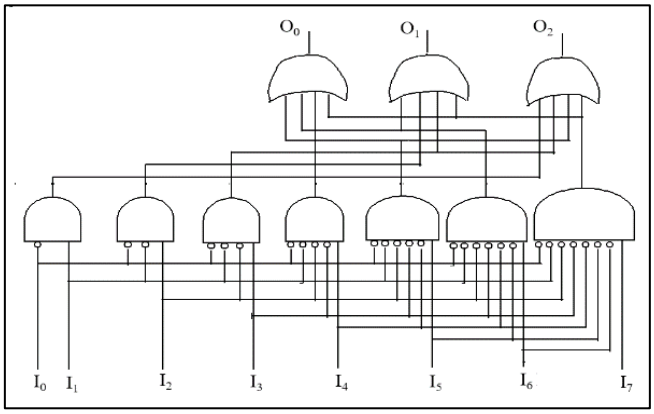

图 2.3　8-3 优先级编码器原理图

实验步骤如下：

1） 根据图 2.3 所示的原理图添加逻辑门。在 Logisim 中添加子电路：8-3 优先级编码器。在工作区中放置与门、或门、输入引脚、分线器、16 进制数码管等。将或门输入端口数属性改为 4，每个与门的输入端口数属性改为原理图所示个数，并修改输入引脚的极性（是否反转）。

2） 添加线路。将输入引脚、逻辑门的输入端、输出端、输出引脚等通过连接线相连。

3） 添加标识符。选中输入/输出引脚，在属性表中添加引脚标识符；选中逻辑门，在属性表中添加门标识符；点击快捷工具栏中文本工具，在电路空白处添加描述文字。

经过以上 3 个步骤后得到如图 2.4 所示的 8-3 优先级编码器电路图。

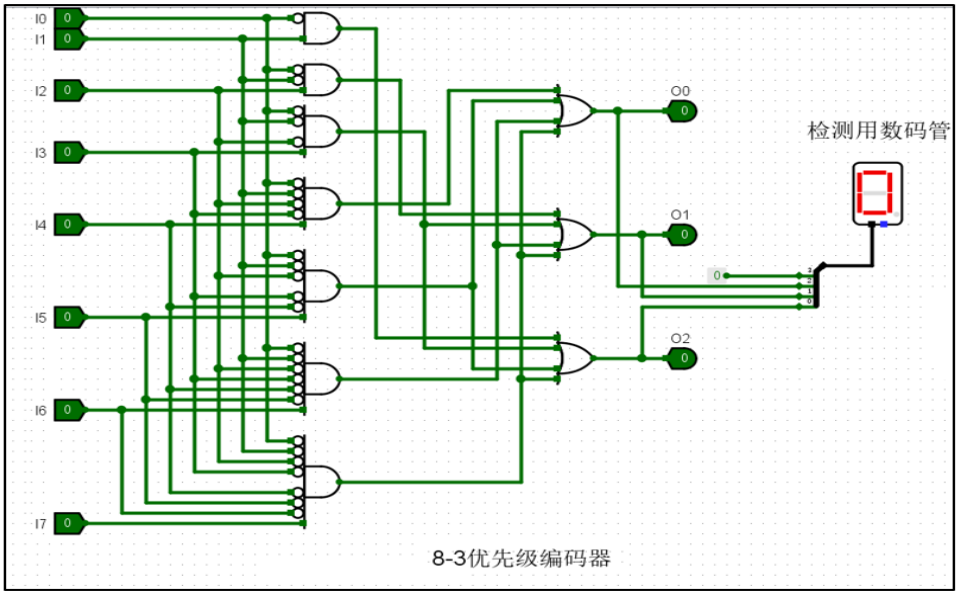

图 2.4　8-3 优先级编码器电路图

4） 仿真测试。进入仿真状态，改变输入引脚赋值，记录输出引脚值，填写表 2.2 所示的优先级编码器功能表，以验证实验结果。保存电路设计文件。

表 2.2　8-3 优先级编码器功能表

输　入								输　出			
I0	I1	I2	I3	I4	I5	I6	I7	Q2	Q1	Q0	Hex 显示
1	X	X	X	X	X	X	X				
0	1	X	X	X	X	X	X				
0	0	1	X	X	X	X	X				
0	0	0	1	X	X	X	X				
0	0	0	0	1	X	X	X				
0	0	0	0	0	1	X	X				
0	0	0	0	0	0	1	X				
0	0	0	0	0	0	0	1				

3.　加法器实验（验收）

串联 4 个全加器子电路实现 4 位串行进位加法器。将加数、被加数和和分别连接到 16 进制数码显示管进行验证。实验步骤如下：

1） 设计全加器。在 Logisim 中添加子电路：全加器。根据全加器逻辑电路图，在 Logisim 工作区中添加逻辑门、连线和标识符。然后将或门输入端口数改为 3；将输入引脚、逻辑门的输入端和输出端、输出引脚等通过连接线相连。选中输入、输出引脚，在属性表中添加引脚标识符。选中逻辑门，在属性表中添加门标识符。点击快捷工具栏中文本工具，在电路空白处添加描述文字；如图 2.5 所示。

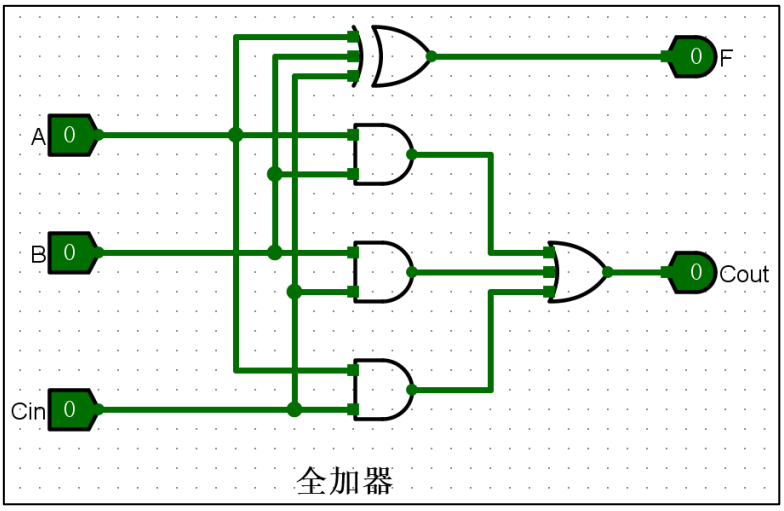

图 2.5 全加器电路图

验证电路功能，保存电路设计文件。

2） 设计 4 位串行进位加法器。在 Logisim 中添加子电路：4 位串行加法器。在 Logisim 工作区中按

图 2.6 所示进行组件布局，其中包含输入输出引脚、隧道、分线器、全加器子电路、常量和 16 进制数字显示等组件。修改组件属性，连接端口，实现 4 位串行进位加法器电路，并验证功能，记录测试数据。保存电路设计文件。

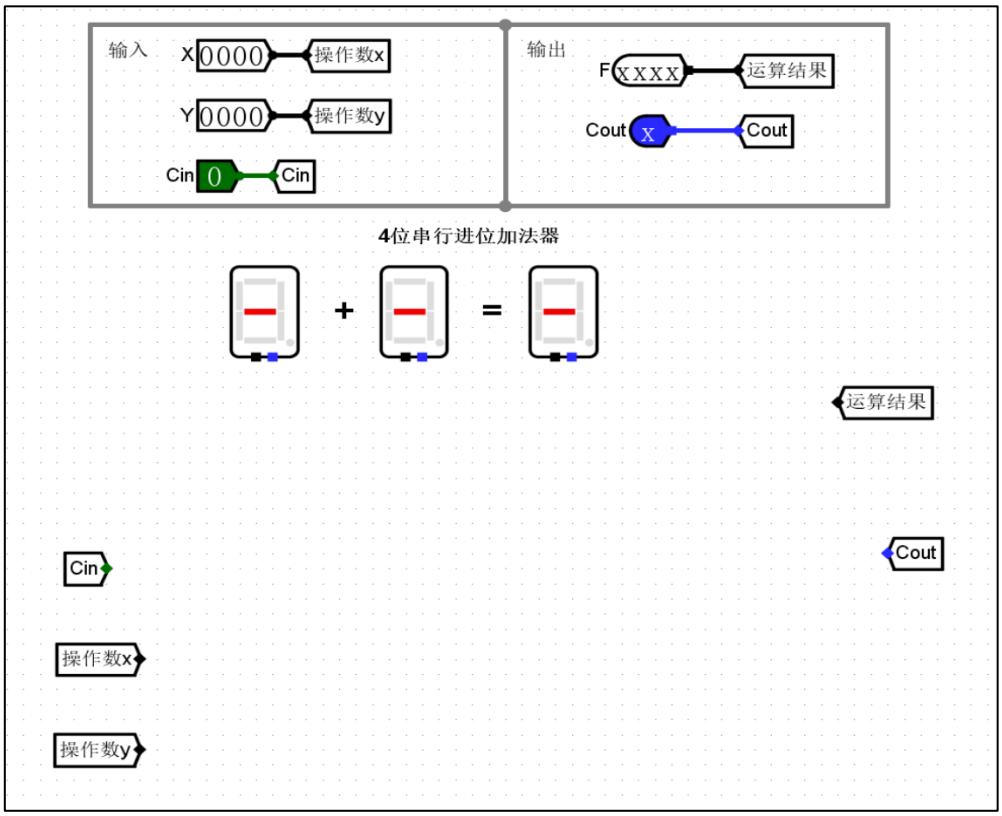

图 2.6　4 位串行加法器组件布局图

3） 设计并实现 4 位加减法器。在 Logisim 中添加子电路：4 位加减器。修改上述电路，添加必要的组件（提示：可使用位扩展组件）。当 Cin=0 时，执行加法运算；当 Cin=1 时，执行减法运算。进行功能验证，并分析结果数据。

4、汉明码校验电路（验收）

数据校验大多采用“冗余校验”的思想，即除原数据信息外，还增加若干位附加的编码，这些新增编码称为校验位。图 2.7 给出了一般情况下的处理过程。

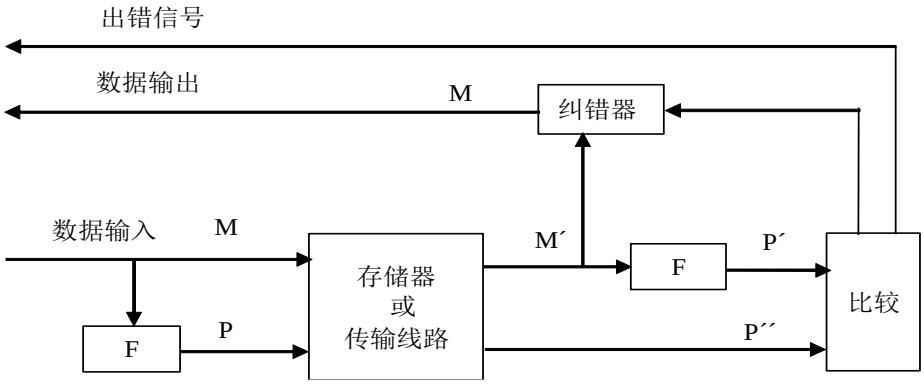

图 2.7 数据校验过程示意图

当数据 M 被存入存储器或从源部件开始传输时，对 M 进行某种运算（用函数 F 来表示），以产生相应的代码 P=F(M)，这里 P 就是校验位。这样原数据信息 M 和相应的校验位 P 一起被存储或传送。当数据被读出或传送到目标部件时，和数据信息一起被存储或传送的校验位也被得到，用于检错和纠错。假定读出后的数据为 M′，通过同样的运算 F 对 M′ 也得到一个新的校验位 P′=F(M′)，假定原来被存储的校验位 P 取出后其值为 P′′，将校验位 P′′ 与新生成的校验位 P′ 进行比较运算，生成一个故障字，根据故障字可以确定是否发生了差错。

最简单的数据检错方法是奇偶校验，通过判断数据 M 中 1 的个数是否发生了奇偶性变化来进行检错。若发生奇偶变化，则故障位 S= P′⊕P′′=1。

汉明码（Hamming Code，也译为海明码）的主要思想是，将数据按某种规律分成若干组，对每组进行相应的奇偶检测，以提供多位校验信息，得到相应的故障字，根据故障字对发生的错误进行定位，并将其纠正。汉明校验码实质上就是一种多重奇偶校验码。

对于只能对单个位出错的情况进行定位和纠错的单纠错码（SEC），进行汉明校验的主要思想如下：将需要进行检/纠错的数据分成 i 组，每组对应 1 位校验位，共有 i 位校验位，因此，故障字为 i 位。若故障字为 0，表示无错；否则故障字的数值就是出错位在码字中的位置编号。除去 0 的情况，i 位故障字的编码个数为 2^i-1，因此构造的码字最多有 2^i-1 位，例如，当 i=3 时，码字可以有 7 位，其中 3 位为校验位，4 位为数据位。为了方便判断码字中出错的是校验位还是数据位，可将校验位的位置编号设为 2 的幂次，即校验位排在第 1（001）、2（010）、4（100）、… 的位置上，其余位置上为数据位。这样，当故障字中只有一位为 1 时，说明是校验位出错，否则就是数据位出错。例如，当 i=3 时，假设校验码为 $P_3P_2P_1$，数据信息为 $M_4M_3M_2M_1$，则码字排列为 $P_1P_2M_1P_3M_2M_3M_4$。通常把上述由数据位和校验位构成的码字称为汉明码。图 2.8 给出了 7 位汉明码的故障字和出错情况的对应关系。

序号 / 分组 含义	1	2	3	4	5	6	7	故障字	正确	出错位
	P_1	P_2	M_1	P_3	M_2	M_3	M_4			1 2 3 4 5 6 7
第3组				√	√	√	√	S_3	0	0 0 0 1 1 1 1
第2组		√	√			√	√	S_2	0	0 1 1 0 0 1 1
第1组	√		√		√		√	S_1	0	1 0 1 0 1 0 1

图 2.9　8 位汉明码的故障字和出错情况对应关系

如图 2.9 所示，第 1 组的故障位 S_1 由校验位 P_1 和数据位 M_1、M_2、M_4 生成，第 2 组的故障位 S_2 由校验位 P_2 和数据位 M_1、M_3、M_4 生成、第 3 组的故障位 S_3 由校验位 P_3 和数据位 M_2、M_3、M_4 生成。假设在终部件得到的数据位 M′为 $M_4M_3M_2M_1$，校验位 P′′为 $P_3P_2P_1$，每组采用偶校验，则根据 M′得到 P′的每一位如下：

$$P_1' = M_1 \oplus M_2 \oplus M_4$$
$$P_2' = M_1 \oplus M_3 \oplus M_4$$
$$P_3' = M_2 \oplus M_3 \oplus M_4$$

因为故障字 S=P′⊕P′′，因此，根据 P′和 P′′得到故障字的每一位如下：

$$S_1 = M_1 \oplus M_2 \oplus M_4 \oplus P_1$$
$$S_2 = M_1 \oplus M_3 \oplus M_4 \oplus P_2$$
$$S_3 = M_2 \oplus M_3 \oplus M_4 \oplus P_3$$

因此，在终部件的汉明码检/纠错电路只要根据在终部件得到的数据位 $M_4M_3M_2M_1$ 和校验位 $P_3P_2P_1$ 形成的码字，按图 2.9 所示的方式划分成 3 组，每组按照上述偶校验方式，得到每一组的故障位 S_i，由故障位构成的故障字 $S_3S_2S_1$ 的值就能确定码字中哪一位发生了错误。

图 2.10 给出了 7 位汉明码检/纠错电路原理图。

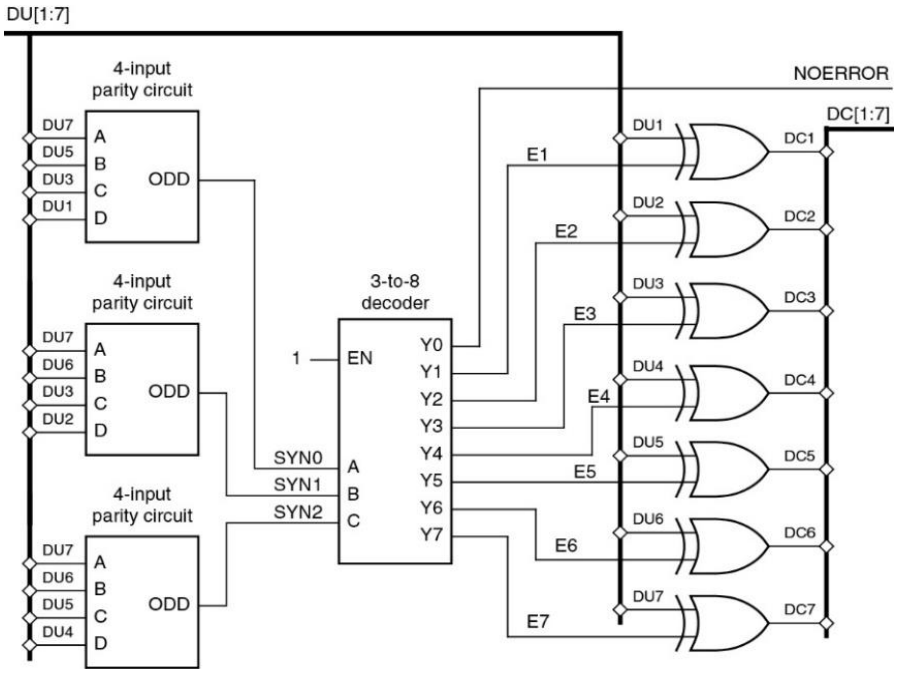

图 2.10　7 位汉明码检/纠错电路原理图

在 Logisim 中添加子电路：7 位汉明码纠错电路。利用译码器、奇偶校验子电路、隧道和分线器等组件实现 7 位汉明码纠错功能，并显示发生错误位置和状态，参考组件布局图如 2.11 所示，输入测试数据并进行验证说明，保存设计文件。

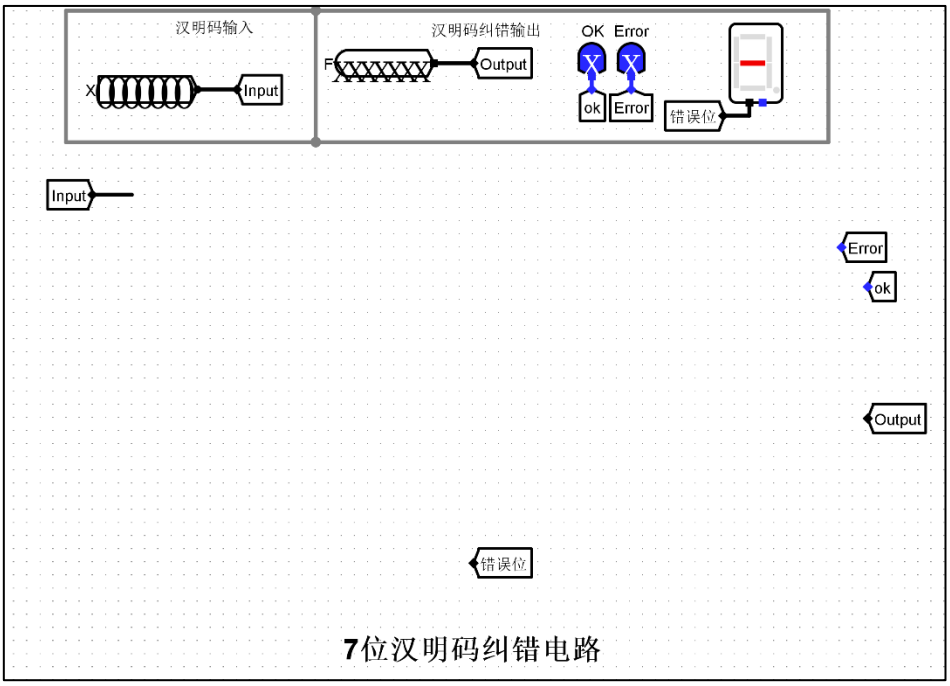

图 2.11　7 位汉明码纠错电路组件布局图

5、桶形移位器（验收）

桶形移位器采用组合逻辑的方式来实现移位功能，能在一个时钟周期内完成多位移动，具有很高的效率，常被用在 ALU 中来实现移位运算。它具有 n 位数据输入和 n 位数据输出，以及指定移动方向、移动类型（算术/逻辑/循环）和移动位数等。

8 位桶形移位器的输入输出引脚图，如图 2-12 所示。其中输入数据 din 和输出数据 dout 均为 8 位，移位位数 shamt 为 3 位。选择端 L/R 表示左移和右移，置为 1 为左移，置为 0 为右移。选择端 A/L 为算术/逻辑选择，置为 1 为算术右移，置为 0 为逻辑右移。

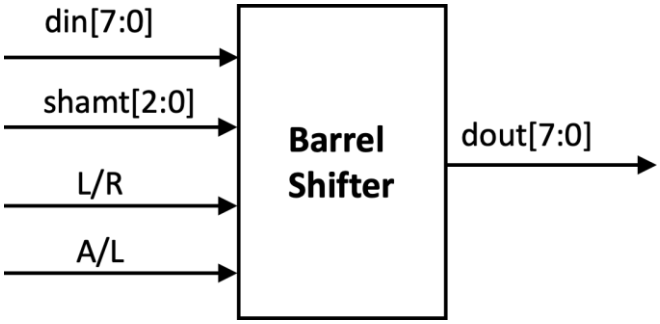

图 2-12　8 位桶形移位器逻辑符号图

该桶形移位器的设计原理图如图 2-13 所示。为了表明数据位的移动方向，该设计使用了大量的 1

位四路选择器来实现（在实验时，可以使用 8 位 4 路选择器来实现），分三级实现 0 至 7 位的左移或右移。第一级利用 shamt[0]来控制是否需要移动一位，第二级在第一级的移动结果上用 shamt[1]来控制是否要移动两位，第三级在第二级的基础上再对应判断是否要移动四位。每个四路选择器有两位控制端，控制端低位 S0 为当前级是否需要移动，对应 shamt[i]，当 S0=0 时，选中 4 路选择器的 0 号或 2 号输入端口，均不做任何移动。当 S0=1 时，控制端高位 S1 对应 L/R 输入，当 S1=0 时，表示右移，选中 4 路选择器的 1 号输入端口；当 S1=1 时，表示左移，选中 4 路选择器的 3 号输入端口。这两个输入端分别连接了数据低位或高位的上一级输出。对于算术和逻辑右移的操作，是通过 A/L 来选择移入的是 0 还是 din[7]。注意，这个电路是纯组合逻辑，所以可以在输入改变时无需时钟信号就直接改变输出的移位结果。

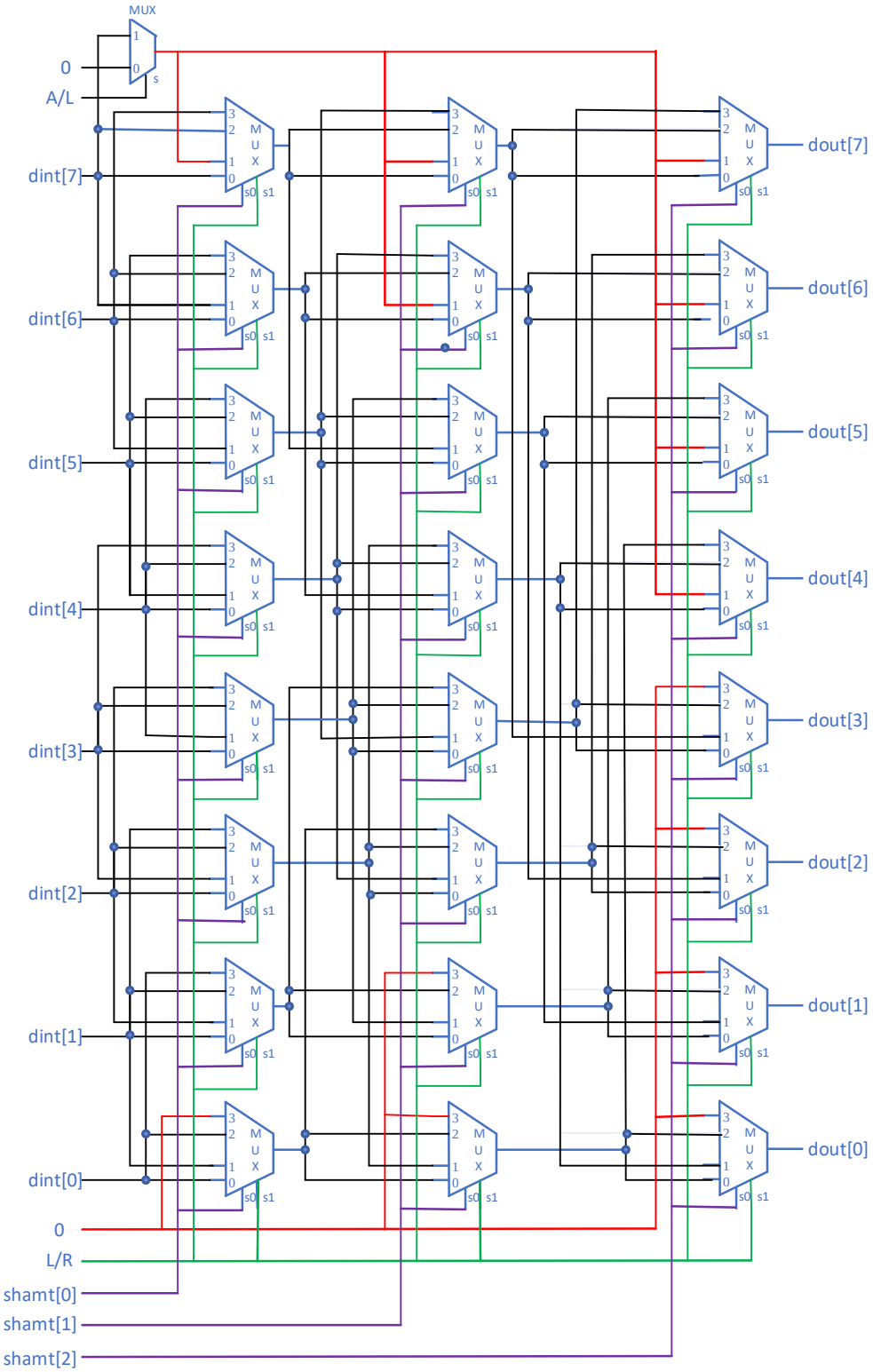

图 2-13 桶形移位器电路原理示意图

在 Logisim 中添加子电路：8 位桶形移位器。利用 8 位 4 路选择器、分线器、常量 0 等组件实现 8 位桶形移位器，参考布局图如 2.14 所示，输入测试数据并进行验证说明，保存设计文件。

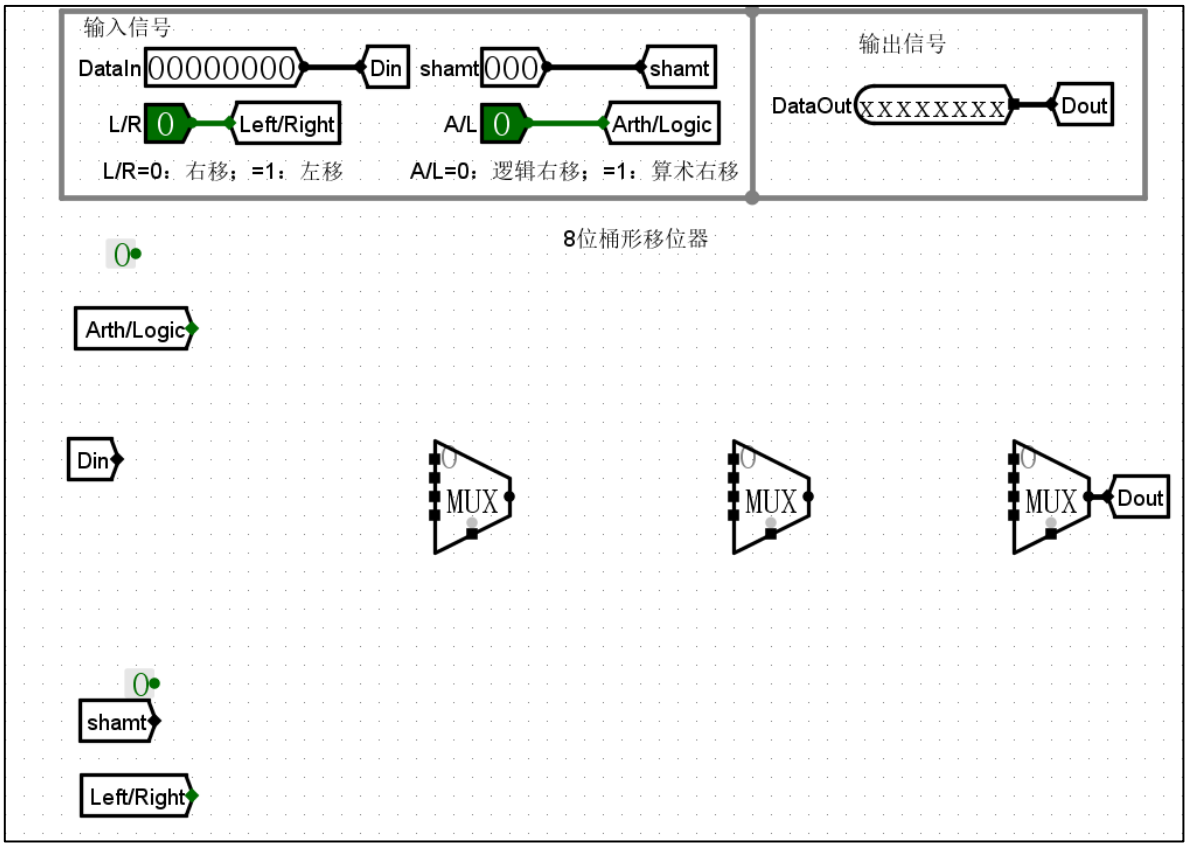

图 2.14　8 位桶形移位器组件布局图

四、思考题

1. 修改实验中的加法器电路，生成进位标志 CF、溢出标志 OF、符号标志 SF 和结果为零标志位 ZF。

2. 在执行比较指令时，通常使用减法运算后，判断标志位的方式来实现，试通过上述加法器实验举例说明判别的方法。

3. 如何使用 8 位桶形移位器扩展到 32 位桶形移位器。

4. Logisim 提供输出组件 LED 矩阵，通过点亮 led 灯的方式显示字符，修改 LED 矩阵行列属性为 16*16，显示“南大”两个汉字，字体可自选。