

实验 4：加法器和 ALU 设计

一、实验目的

1. 掌握先行进位部件 CLU 和先行进位加法器 CLA 的设计方法。
2. 掌握 32 位先行进位加法器的设计方法。
3. 掌握 ALU 的设计方法。

二、实验环境

Logisim: <https://github.com/Logisim-Ita/Logisim>

三、实验内容

1. 4 位先行进位加法器 CLA

几乎所有算术运算都要用到 ALU 或加法器，而 ALU 的核心还是加法器，因此要提高计算机的运算速度，关键在于提升加法器的执行速度。为了提高加法器的速度，必须尽量避免进位之间的依赖关系和传递关系，先行进位（也称超前进位）部件（Carry Lookahead Unit, CLU）就能实现这一功能。

把进位传递函数 $P_i = X_i + Y_i$ 和进位生成函数 $G_i = X_i Y_i$ 列入到 4 位二进制数加法器进位 $C_1 \sim C_4$ 的逻辑表达式中，可以得到以下 4 个先行进位 C_i 的逻辑表达式，从公式(4-1)中的表达式可以看出， C_i 仅与 X_i 、 Y_i 和 C_0 有关，相互间的进位没有依赖关系。只要 $X_1 \sim X_4$ 、 $Y_1 \sim Y_4$ 和 C_0 同时到达，就可几乎同时形成 $C_1 \sim C_4$ ，并同时生成各个数位的和。

$$\left. \begin{aligned} C_1 &= G_0 + P_0 C_0 \\ C_2 &= G_1 + P_1 G_0 + P_1 P_0 C_0 \\ C_3 &= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0 \\ C_4 &= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0 \end{aligned} \right\} \quad (4-1)$$

4 位先行进位部件 CLU 设计原理图如 4.1 所示。

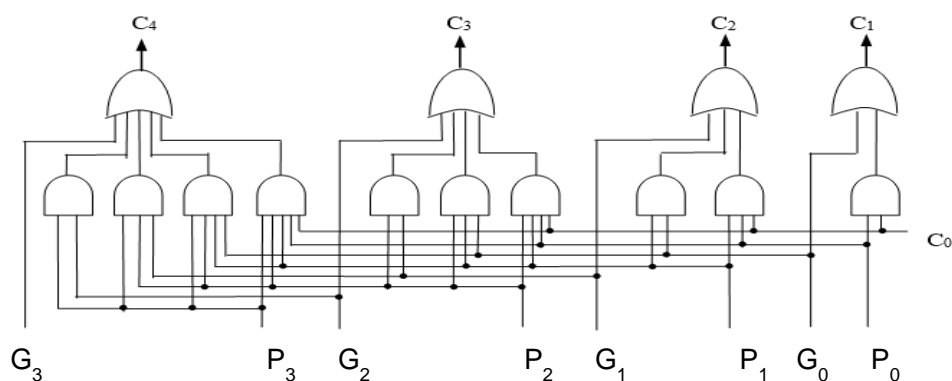


图 4.1 4 位 CLU 原理图

更多位数的加法器可通过分组的方式来实现，采用组内和组间都并行的进位方式。为了实现组间并行，需要在先行进位部件中输出组间进位生成函数 G_g 和组间进位传递函数 P_g 。

$$P_g = P_3 P_2 P_1 P_0$$

$$G_g = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$

将式(4-1)中进位 C_4 的逻辑表达式改写成为 $C_4 = G_g + P_g C_0$ ，CLU 部件中增加两个输出端口 P_g 和 G_g 。

实验步骤如下：

1) 设计 4 位先行进位部件 CLU。在 Logisim 中添加子电路：4 位先行进位部件。在 Logisim 工作区中按图 4.2 所示，定义输入输出引脚和组件布局，其中包含输入输出引脚、隧道、逻辑门电路等组件。修改组件属性，连接端口，实现 4 位先行进位部件电路。通过对 P_i 和 G_i 设置不同的输入值，观察 C_i 的输出值，以验证电路的正确性，记录测试数据。封装子电路如图 4.3 所示。保存电路设计文件。

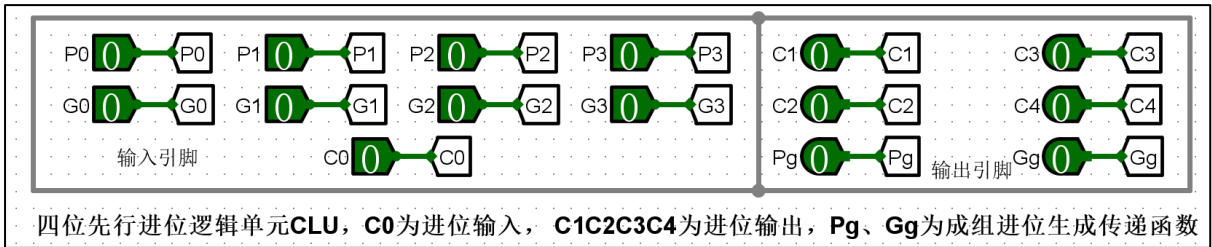


图 4.2 4 位先行进位部件组件布局图

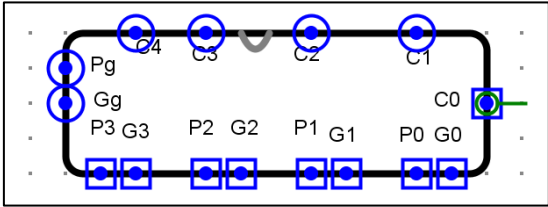


图 4.3 4 位先行进位部件子电路封装图

2) 设计支持进位生成函数和进位传递函数的全加器。在 Logisim 中添加子电路：全加器。在 Logisim 工作区中按图 4.4 所示，定义输入输出引脚和组件布局。改变输入数据，记录测试数据，验收电路功能。封装子电路如图 4.5 所示。保存电路设计文件。

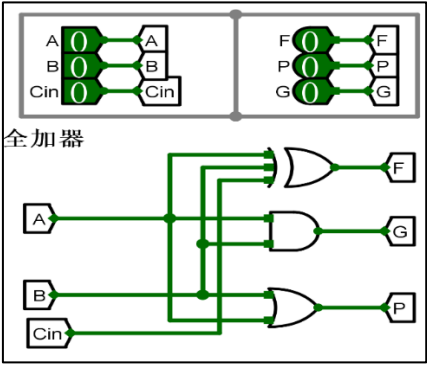


图 4.4 支持进位生成和传递函数的全加器电路图

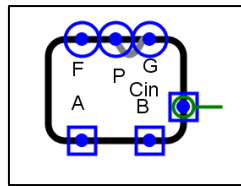


图 4.5 全加器子电路封装图

3) 4 位快速加法器 CLA 实验。在 Logisim 中添加子电路：4 位快速加法器。在 Logisim 工作区中按图 4.6 所示，定义输入输出引脚和组件布局，其中包含输入输出引脚、隧道、4 位先行进位部件子电路、全加器子电路、LED 数码管、逻辑门电路等组件。为了能够实现组间并行运算，增加了组间进位产生函数 Gg 和组间进位传递函数 Pg 的输出。修改属性，参照课件和教材上的电路原理图，连接组件，实现电路。通过设置不同的 X 、 Y 和 Cin 的输入值，观察结果 F 、进位 $Cout$ 、 Pg 、 Gg 等输出值，以验证电路的正确性，记录测试数据。封装子电路如图 4.7 所示。保存电路设计文件。

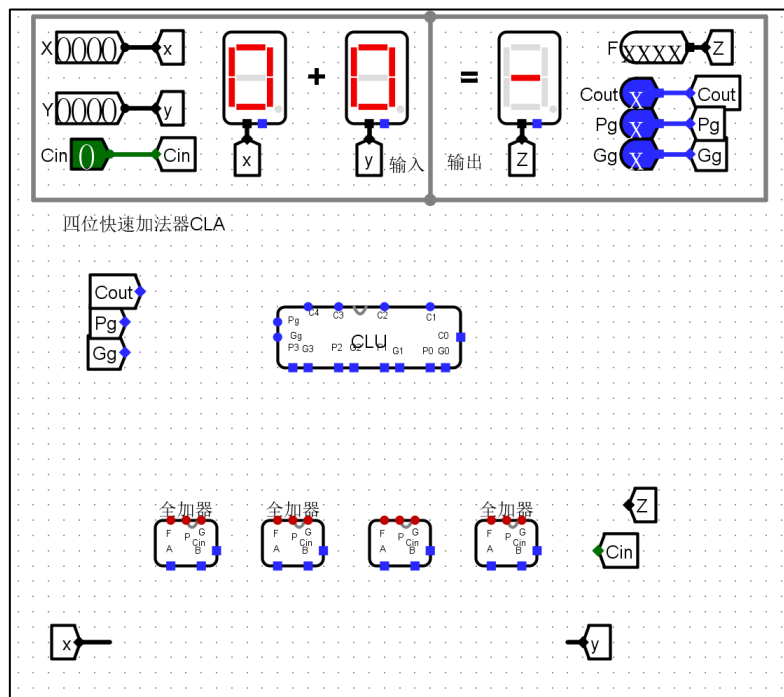


图 4.6 4 位快速加法器组件布局图

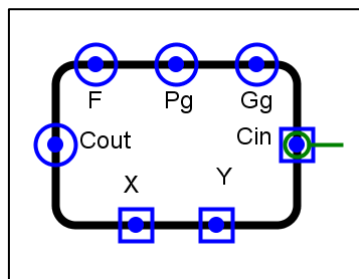


图 4.7 4 位快速加法器子电路封装图

2. 16 位两级先行进位加法器实验

对于一个 16 位加法器，可以分成 4 组，每组用一个 4 位先行进位加法器 CLA 实现。图 4.8 是一个由 4 个 4 位先行进位加法器 CLA 与一个组间先行进位部件 CLU 构成的 16 位两级先行进位加法器原理图。

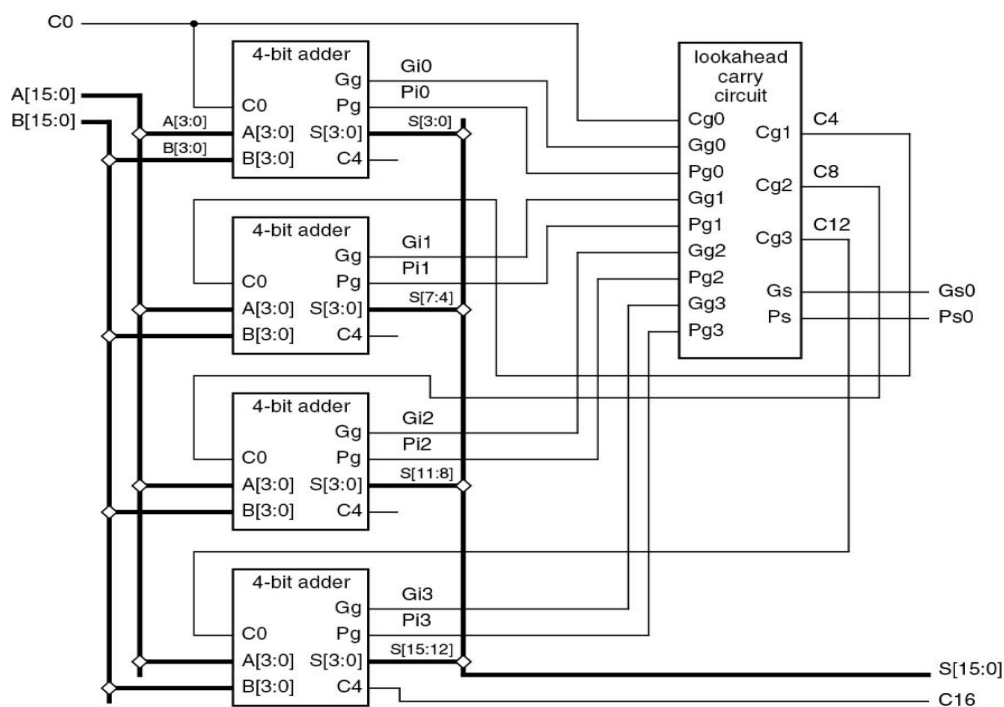


图 4.8 16 位两级先行进位加法器原理图

本实验要求根据图 4.8 所示的原理图，设计实现一个 16 位两级先行进位加法器电路，并对加法器的功能进行验证。

实验步骤如下。

1) 在 Logisim 中添加子电路：16 位先行进位加法器。在 Logisim 工作区中按图 4.9 所示，定义输入输出引脚和组件布局，其中包含输入输出引脚、隧道、4 位快速加法器、4 位先行进位部件子电路、逻辑门电路等组件。参照图 4.8 的电路原理图，连接组件，实现电路。通过设置不同的 X、Y 和 Cin 的输入值，观察结果 F、进位 Cout 等输出值，以验证电路的正确性，记录测试数据。封装子电路如图 4.10 所示。保存电路设计文件。

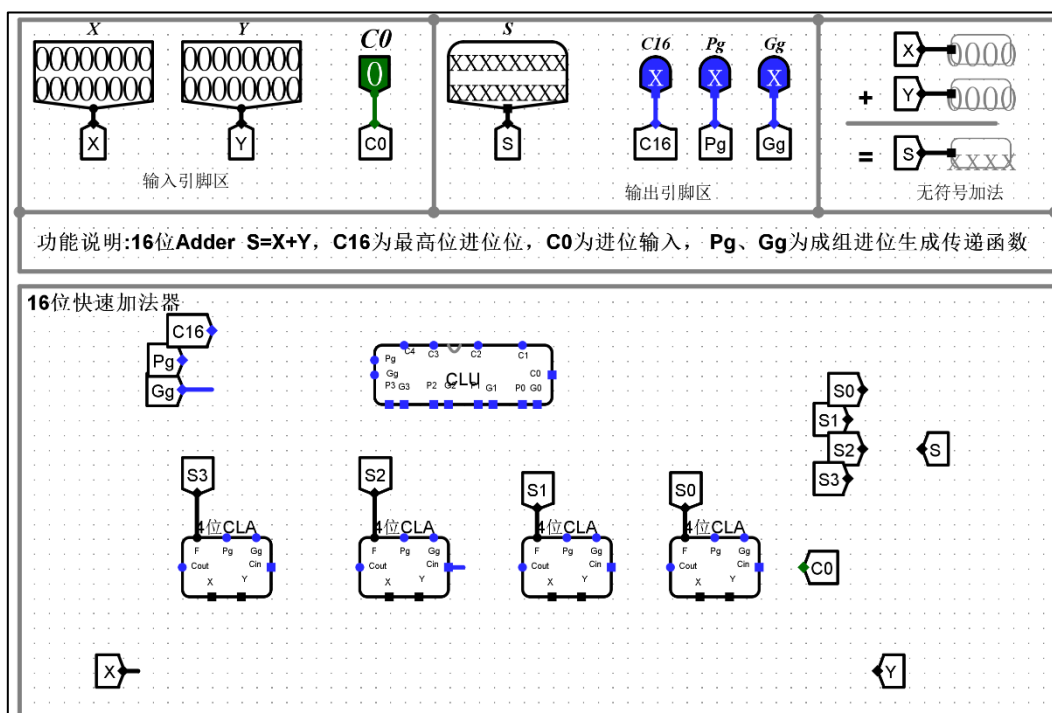


图 4.9 16 位两级先行进位加法器布局图

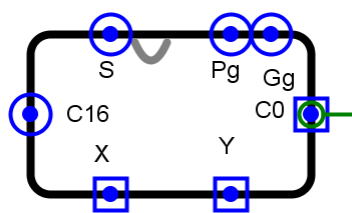


图 4.10 16 位两级先行进位加法器封装图

3. 32 位加法器构建实验（验收）

通过将两个 16 位两级先行进位加法器串行级联构建一个 32 位加法器，并根据给出的标志位生成电路原理图（如图 4.11 所示），在 32 位加法器中生成 CF、SF、OF、ZF 标志位。

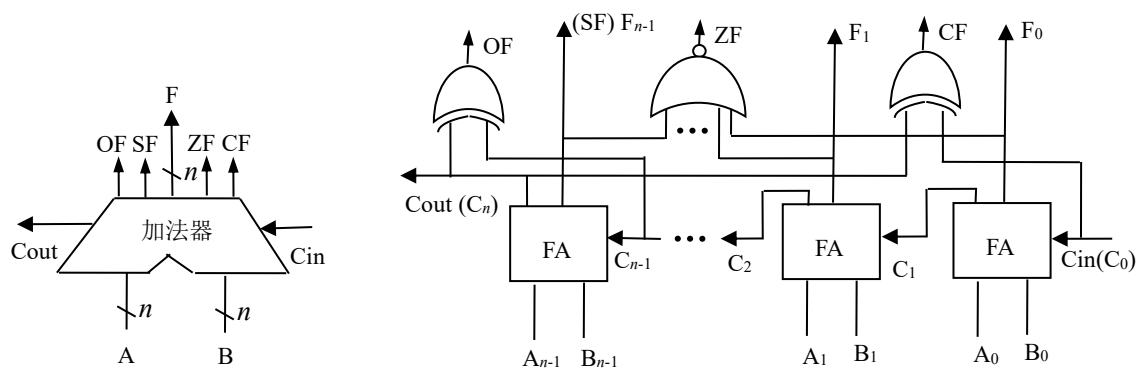


图 4.11 加法器标志位生成电路图

溢出标志位 OF 可用公式 $OF = \overline{A_{n-1}} \cdot \overline{B_{n-1}} \cdot F_{n-1} + A_{n-1} \cdot B_{n-1} \cdot \overline{F_{n-1}}$ 来实现。结果为 0 标志位 ZF 的可

采用分组分级进行或运算的方式获取，避免扇入输入数量过多的问题。

实验步骤如下。

1) 创建子电路。在 Logisim 中添加一个名为“32 位加法器”的子电路，并双击该子电路名称，在右侧工作区中构建相应电路。

2) 设计子电路并进行功能测试。如图 4.12 所示，在工作区中添加两个 16 位先行进位加法器、逻辑门、分线器、输入/输出引脚、探针等部件，将它们布局到适当位置，进行线路连接，添加标识符和电路功能描述文字。通过设置加数 A、B 和低位进位 Cin 的输入值，观察相加和 F、标志位和向高位的进位 Cout 等输出值，记录测试数据，验证电路的正确性。封装子电路如图 4.13 所示。保存电路设计文件。

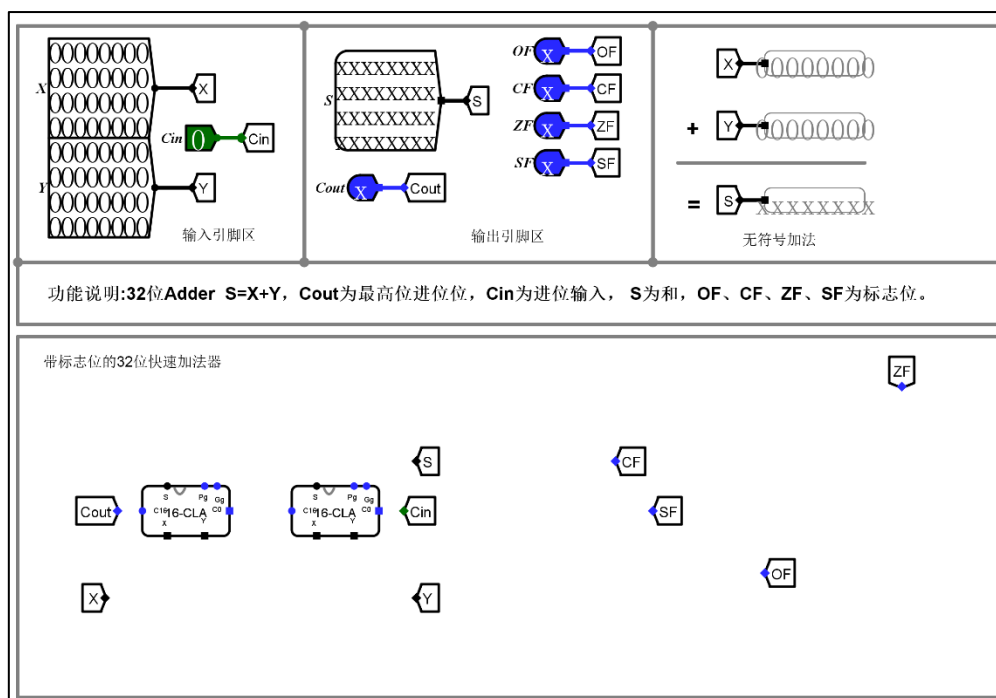


图 4.12 32 位加法器布局图

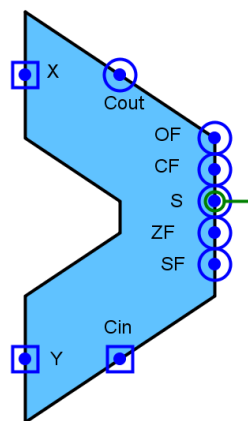


图 4.13 32 位加法器子电路封装图

4. ALU 设计实验（验收）

ALU 是 CPU 中的核心数据通路部件之一，它主要完成 CPU 中需要进行的算术逻辑运算。RV32I 包括 4 种 ALU 指令：比较运算、移位运算、逻辑运算和算术运算。比较运算通过操作数相减然后判断标志位来生成结果；移位运算通过桶形移位器来实现左移和右移；逻辑运算可直接通过逻辑门阵列来实现；算术运算通过加法器以及生成的标志位来实现。ALU 在操作控制信号 ALUctr 的指示下，执行相关运算，Result 作为 ALU 运算的结果被输出，零标志 Zero 被作为 ALU 的结果标志信息输出。

通过对 RV32I 的指令分析，执行这些指令时，主要完成算术运算（加、减）、移位运算（左右移位、算术和逻辑移位）、逻辑运算（与、或、异或）、小于比较置数运算（带符号数和无符号数）、读取操作数 B 等操作。因此 ALU 的操作需要通过输入 ALUctr 信号生成 ALU 控制信号来确定，控制信号包括：SUBctr 用来控制 ALU 执行加法还是减法运算，当 SUBctr=1 时，做减法，当 SUBctr=0 时，做加法；SIGctr 信号控制 ALU 是执行“带符号整数比较小于置 1”还是“无符号数比较小于置 1”功能，当 SIGctr=1，则执行“带符号整数比较小于置 1”，当 SIGctr=0，则执行“无符号数比较小于置 1”；SFTctr 用来控制桶形移位器移位方向，当 SFTctr=0，则执行右移操作，当 SFTctr=1，则执行左移操作；ALctr 用来控制桶形移位器执行算术移位还是逻辑移位，当 ALctr=0，则执行逻辑移位，当 ALctr=1，则执行算术移位；OPctr[2:0]用来控制选择哪种运算的结果作为 Result 输出，因为有加法器和、按位或、按位与、按位异或、移位器输出、操作数 B 输出、小于置 1 等 7 种运算，所以输出结果的多路选择器的选择信号 OPctr 需要 3 位二进制数。图 4.14 给出一个实现 RV32I 指令中运算的 ALU 原理图。

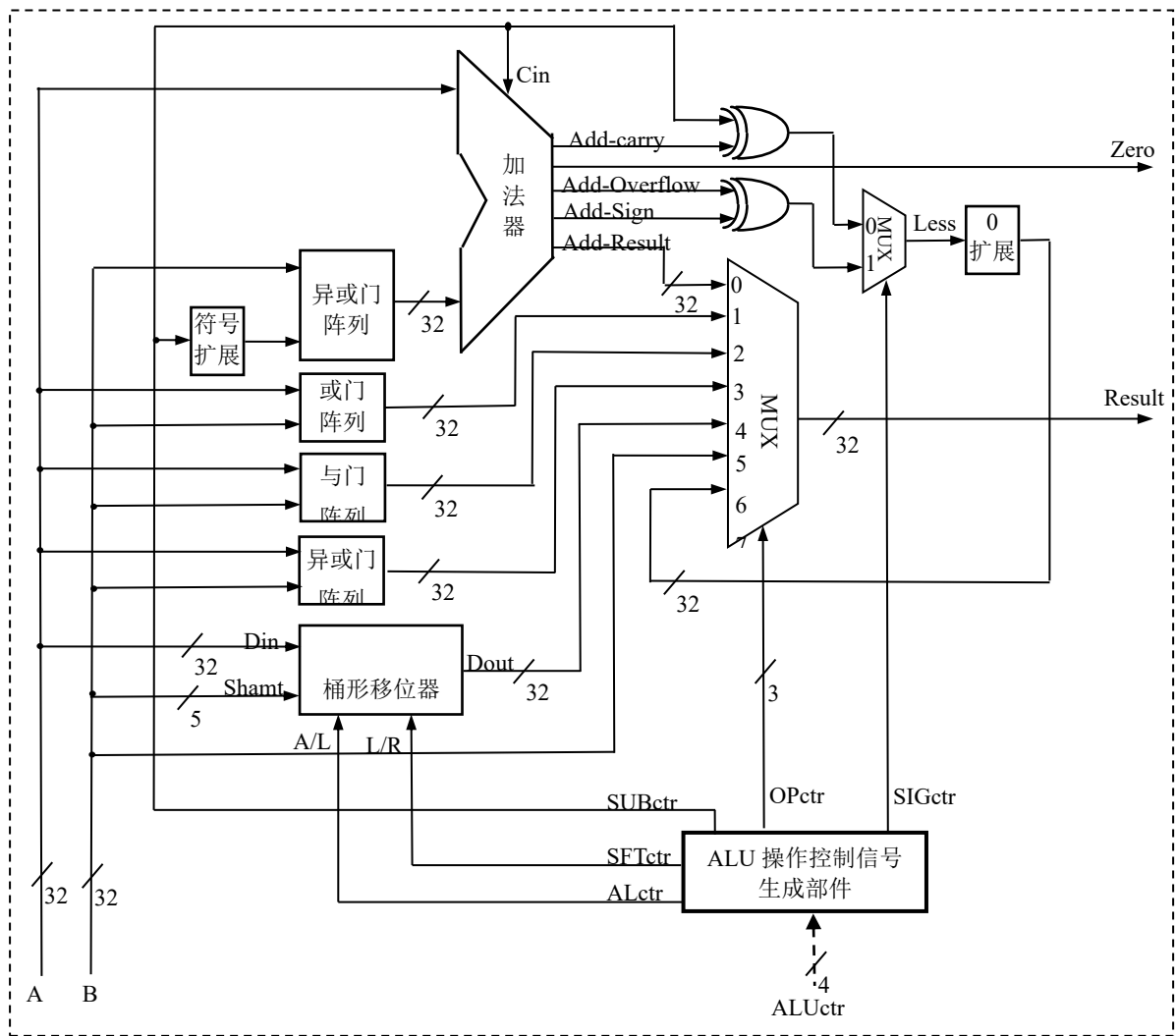


图 4.14 ALU 原理图

ALU 的操作控制与 RISC-V 指令中的 funct3 字段存在较强关系，因此，在对 ALUctr 进行编码时，可以根据 funct3 字段进行优化，使得后续控制器的设计变得更简单。表 4.1 给出了 ALUctr 的一种四位编码方案。

表 4.1 ALUctr 的四位编码及其对应的控制信号

ALUctr<3:0>	指令操作类型	SUBctr	SIGctr	ALctr	SFTctr	OPctr<2:0>	OPctr 的含义
0 0 0 0	auipc, addi, add, jal, jalr, lb, lh, lw, lbu, lhu, sb, sh, sw: 加法	0	×	×	×	000	选择加法器的结果输出
0 0 0 1	sll, slli: 逻辑左移	×	×	0	1	100	选择移位器结果输出
0 0 1 0	slt, slti, beq, bne, blt, bge: 带符号数小于比较	1	1	×	×	110	选择小于置位结果输出
0 0 1 1	sltu, sltui, bltu, bgeu: 无符号数小于比较	1	0	×	×	110	选择小于置位结果输出

0 1 0 0	xor, xori: 异或	×	×			011	选择“按位异或”结果输出
0 1 0 1	srl, srli: 逻辑右移	×	×	0	0	100	选择移位器结果输出
0 1 1 0	or, ori: 或运算	×	×	×	×	010	选择“按位或”结果输出
0 1 1 1	and, andi: 与运算	×	×	×	×	001	选择“按位与”结果输出
1 0 0 0	sub: 减法	1	×	×	×	000	选择加法器的结果输出
1 1 0 1	sra, srai: 算术右移	×	×	1	0	100	选择移位器结果输出
其余	(未用)						
1 1 1 1	lui: 取操作数 B	×	×	×	×	101	选择操作数 B 直接输出

根据表 4.1 可以得到各输出控制信号的逻辑表达式，考虑到 ALUctr 编码的可扩展性，控制信号 SUBctr、SIGctr、ALctr、SFTctr、Opctr[2:0]对应的逻辑表达式使用最小项进行表示，并按照最小风险法进行化简，或者不化简，使用负逻辑输入信号。控制信号的最小项表达式如下：

$$\begin{aligned}
\text{SUBctr} &= \sum m(2, 3, 8) \\
\text{SIGctr} &= m_2 \\
\text{ALctr} &= m_{13} \\
\text{SFTctr} &= m_1 \\
\text{Opctr}[2] &= \sum m(1, 2, 3, 5, 13, 15) \\
\text{Opctr}[1] &= \sum m(2, 3, 4, 6) \\
\text{Opctr}[0] &= \sum m(4, 7, 15)
\end{aligned}$$

根据图 4.14 所示的 ALU 原理图可知，组成 32 位 ALU 的基本部件除了基本的逻辑门、多路选择器和扩展器外，较复杂的子电路包括一个 32 位加法器、32 位移位器和一个 ALU 操作控制部件。前述实验已经设计了一个 32 位加法器并封装生成了相应的子电路，因此，还需要设计一个 ALU 操作控制部件和 32 位桶形移位器，并将其封装成子电路。

ALU 设计实验步骤如下：

1) ALU 操作控制部件实验。在 Logisim 中添加一个名为“ALU 操作控制部件”的子电路，根据控制信号逻辑表达式，按照图 4.15 的组件布局图，在工作区中放置合适的逻辑门电路、多路选择器等组件。设置相应属性，进行线路连接，添加标识符和电路功能描述文字。改变输入数据，验证控制信号输出值，记录测试数据，验证电路的正确性。封装子电路如图 4.16 所示，保存电路设计文件。

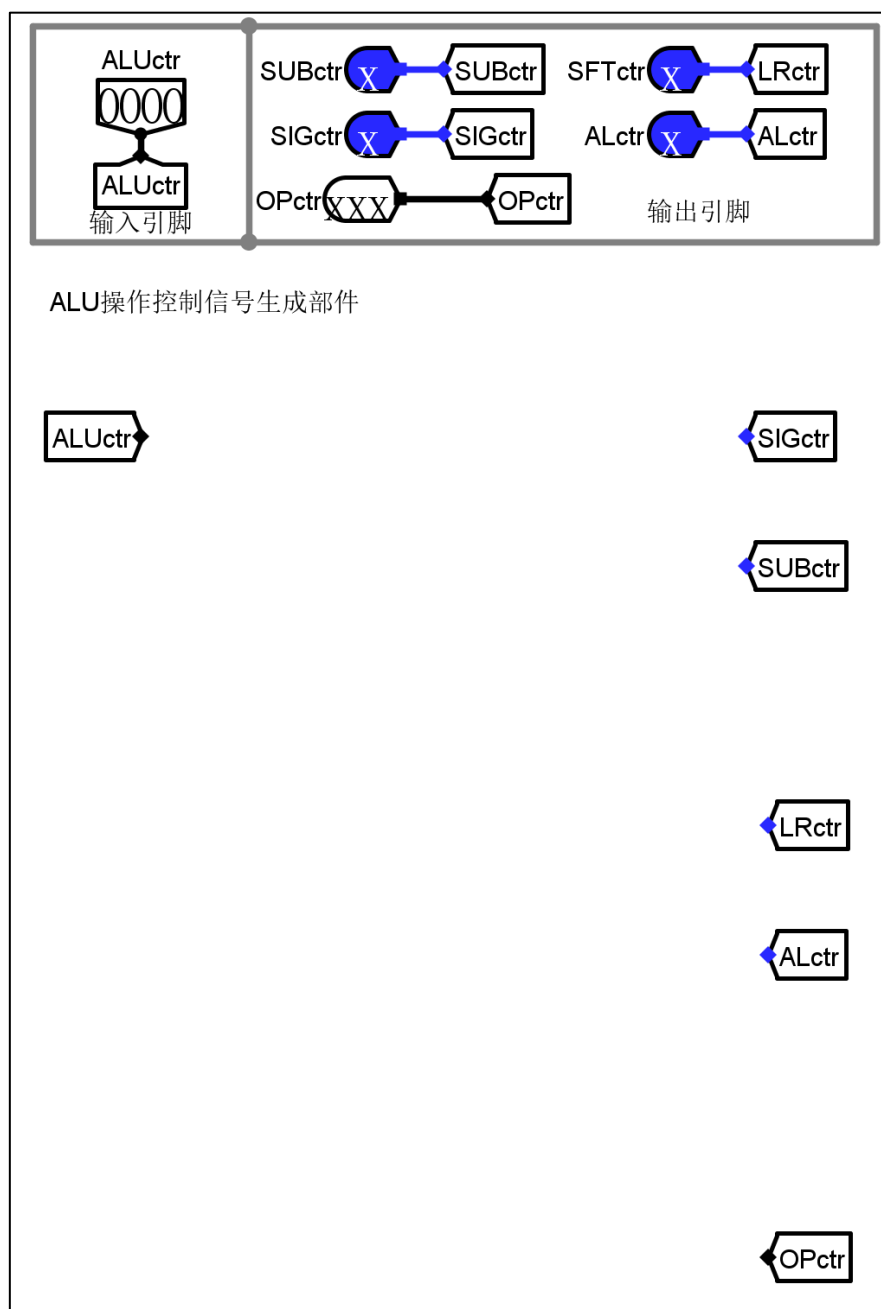


图 4.15 ALUctr 操作控制部件布局图

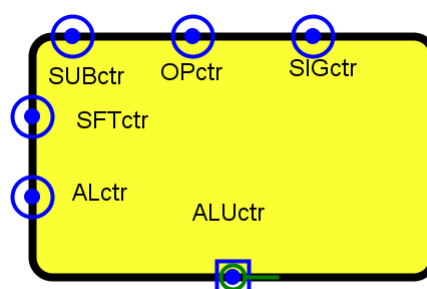


图 4.16 ALUctr 操作控制部件子电路封装图

2) 32 位桶形移位器实验。在 Logisim 中添加一个名为“32 位桶形移位器”的子电路，根据实验 2 中 8 位桶形移位器的电路图，通过扩展多路选择器的输入数据位宽到 32 位，并增加两级 2 路选择器，来实现 32 位移位器。移动位数 shamt 输入端扩展到 5 位，高两位分别表示移动 8 位和 16 位。按照图 4.17 的组件布局图，在工作区中放置合适的逻辑门电路、分线器、位扩展器（提示：同时移位多位相同数字时，可以使用位扩展器来实现）等组件，设置相应属性，进行线路连接，添加标识符和电路功能描述文字。改变输入数据，验证输出信号的输出值，记录测试数据，验证电路的正确性。封装子电路如图 4.18 所示，保存电路设计文件。

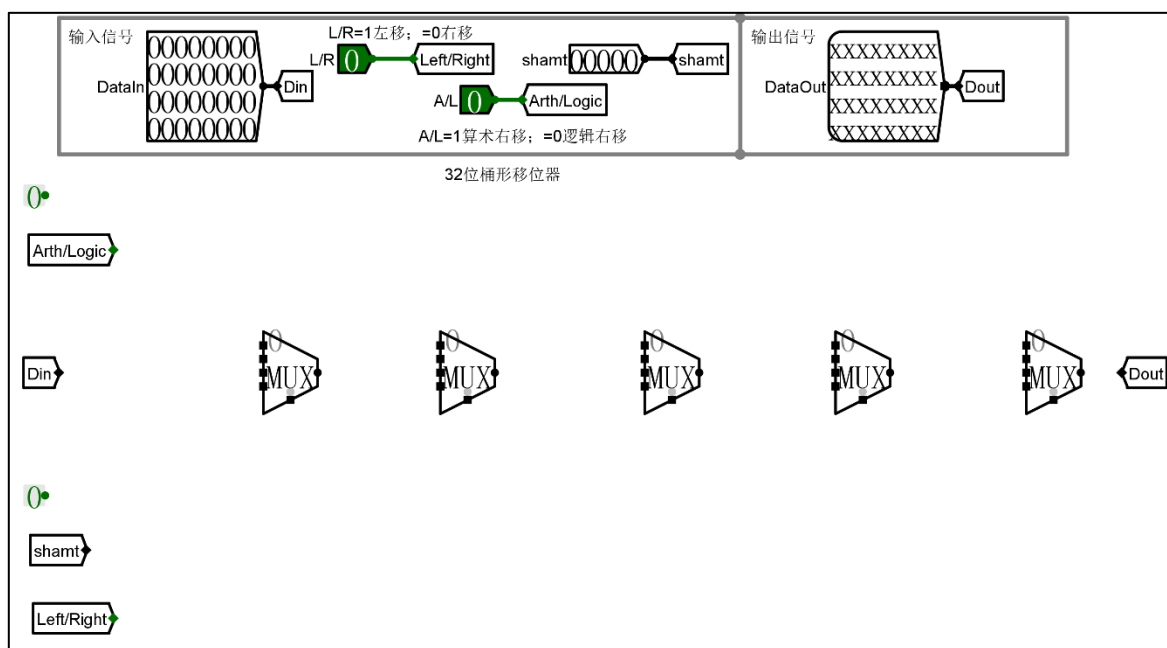


图 4.17 32 位桶形移位器布局图

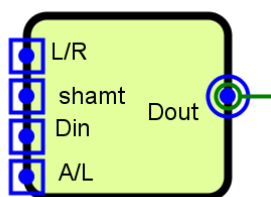


图 4.18 32 位桶形移位器子电路封装图

3) 实现 32 位 ALU。在 Logisim 中添加一个名为“32 位 ALU”的子电路，在右侧工作区中构建相应电路。参照图 4.14 所示的 ALU 原理图，按照图 4.19 所示的 ALU 组件布局图，在工作区中放置所需的 32 位加法器、ALU 操作控制部件、32 位移位器、逻辑门阵列、多路选择器、位扩展器、探针等组件，设置相应属性，进行线路连接，添加标识符和电路功能描述文字，移位器的移动位数由操作数 B 的最低 5 位数决定。改变输入数据，验证输出信号的输出值，记录测试数据，验证电路的正确性。ALU 输入信号包括两个 32 位操作

数 A、B 和一个 4 位 ALU 控制信号 ALUctr，输出信号包括一个 32 位结果数据 S 和一个 1 位结果为 0 的标志位 Zero，封装子电路如图 4.20 所示，保存电路设计文件。提示：由于 32 位加法器直接输出了 CF 标志位，因此不需要将 CIN 和 Adder-carry 异或来生成 CF，而是直接将加法器的 CF 输出端连到 2 选 1 多路选择器的输入端。

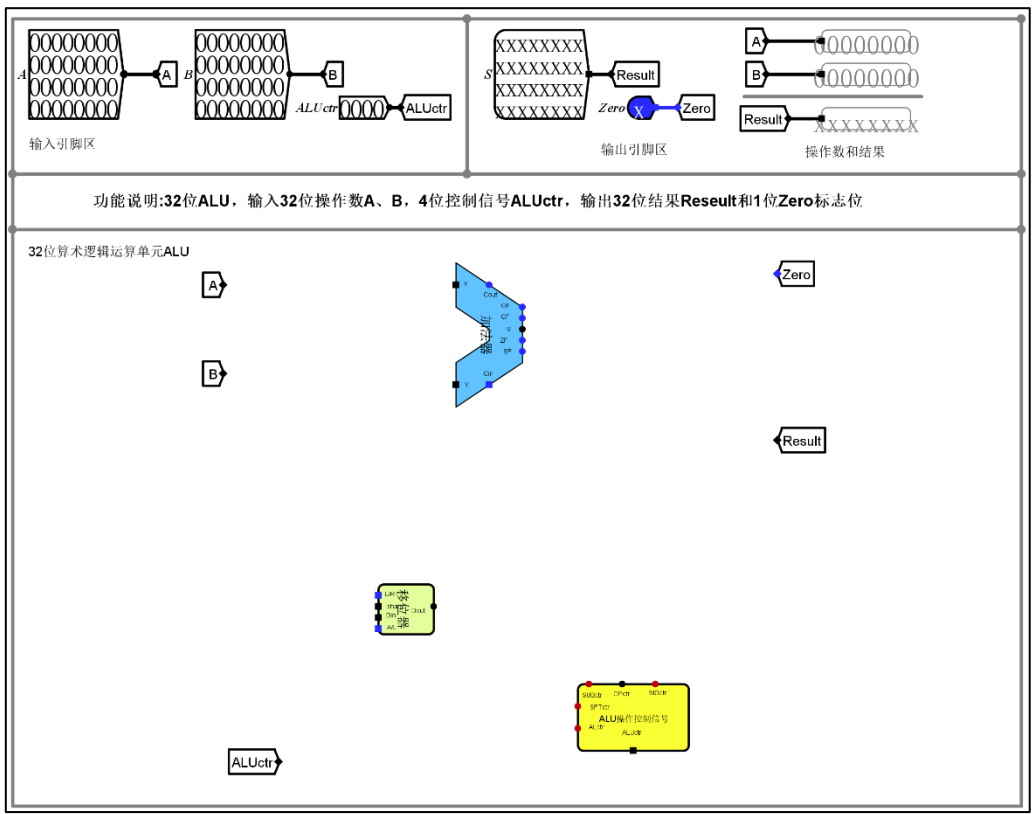


图 4.19 ALU 组件布局图

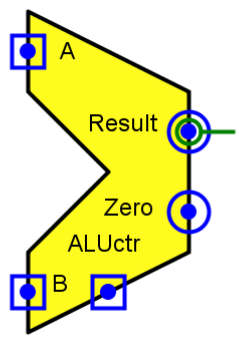


图 4.20 ALU 子电路封装图

4) 功能测试。按照表 4.1 中 ALUctr 编码的定义，分别对 ALUctr 设置不同的值来设定 ALU 的操作功能，对操作数 A 和 B 设置不同的输入值，记录输出数据，观察信号变化，测试并验证操作功能。

四、思考题

1. 利用设计的 ALU，举例说明如何实现一个 32 位整数和一个常量的乘法运算。
2. 查阅资料，说说除了先行进位加法器以外还有哪些高速并行多位二进制数加法算法，至少简要说明一种算法的实现流程。
3. 简要分析加法运算、比较运算和移位运算三种运算中操作数在 ALU 中经过的逻辑门级数。