实验一 组合逻辑设计

**一、实验目的**

1 掌握Verilog语言和Vivado、Logisim开发平台的使用；

2 掌握基础组合逻辑电路的设计和测试方法。

**二、实验内容（用Logisim或Vivado实现）**

1 基础门电路（多输入门电路、复用器等）的设计和测试；

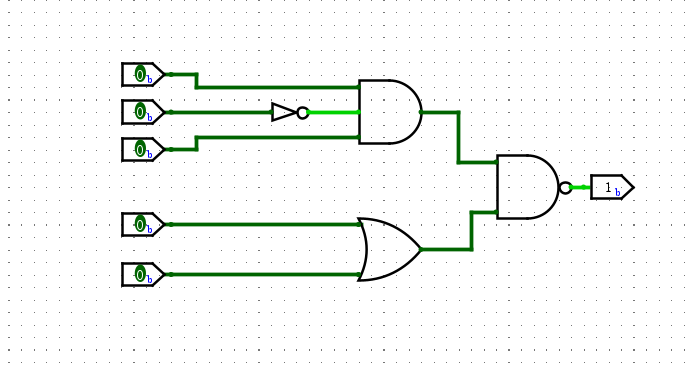
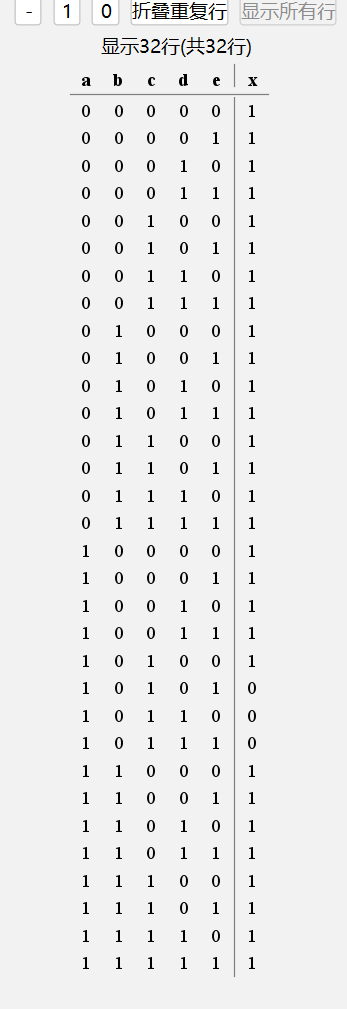
2 基础功能模块（编码器、译码器等）的设计与测试。

**三、实验要求**

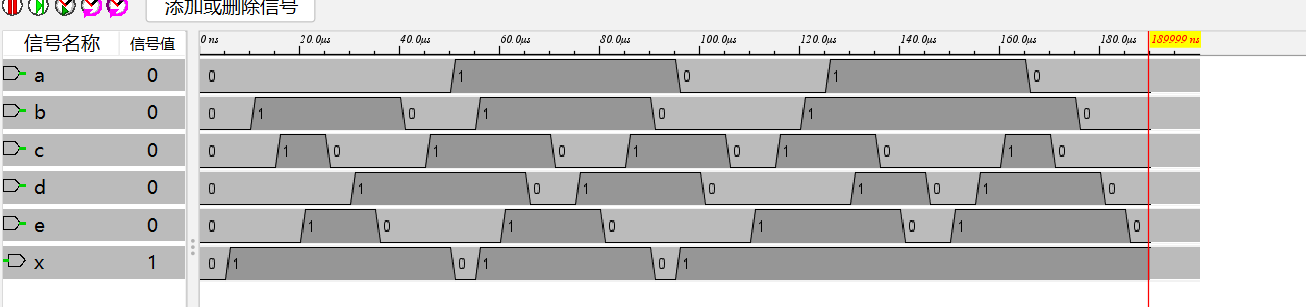
1 掌握Vivado与Logisim开发工具的使用，掌握以上电路的设计和测试方法；

2 记录设计和调试过程（Verilog代码/电路图/表达式/真值表，Vivado仿真结果，Logisim验证结果等）；

3 分析Vivado仿真波形/Logism验证结果，注重输入输出之间的对应关系。

**四、实验过程及分析**

多输入门电路电路图：

****

**五、调试和心得体会**

以上是用logisim实现的多输入门电路，根据这个表达式x=~(a⋅~b⋅c⋅(d+e))所做。

我们可以根据表达式来构建电路：根据表达式，我们可以看出这是一个由四个输入变量 a、b、c、d 和 e 组成的布尔表达式，其中 ~ 表示取反操作，⋅表示逻辑与操作，+表示逻辑或操作。我们可以使用一个与门来实现 a⋅~b⋅c 这个部分，再通过一个非门来实现 ~b。最后，我们可以使用一个或门来实现 (d+e) 这个部分，然后将其与 (a⋅~b⋅c) 通过一个与非门进行逻辑与操作，即可得到最终的结果 x。

也可以通过输入表达式，直接构建电路。

通过真值表可以看到，只有a=1，b=0，c=1，d+e=1时最后的输出x才是为0.

复用器：

Logisim:

图片包含 表格

描述已自动生成文本

描述已自动生成

图示

描述已自动生成

当S为逻辑0（低电平）时，~S为逻辑1（高电平），所以第一项D0⋅~S为D0⋅1=D0，相应地，第二项D1⋅S为D1⋅0=0。因此，当S为逻辑0时，Y的取值为D0。

当S为逻辑1（高电平）时，~S为逻辑0（低电平），所以第一项D0⋅~S为D0⋅0=0，相应地，第二项D1⋅S为D1⋅1=D1。因此，当S为逻辑1时，Y的取值为D1。

首先，使用非门（或称为反向器）将输入信号S进行反相得到~S。

然后，将D0和~S作为输入连接到一个与门中。

同时，将D1和S作为输入连接到一个与门中。

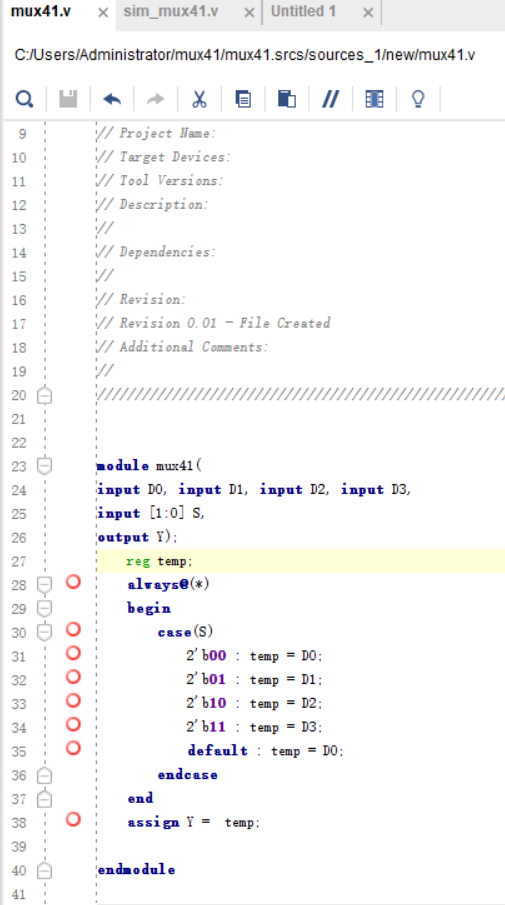
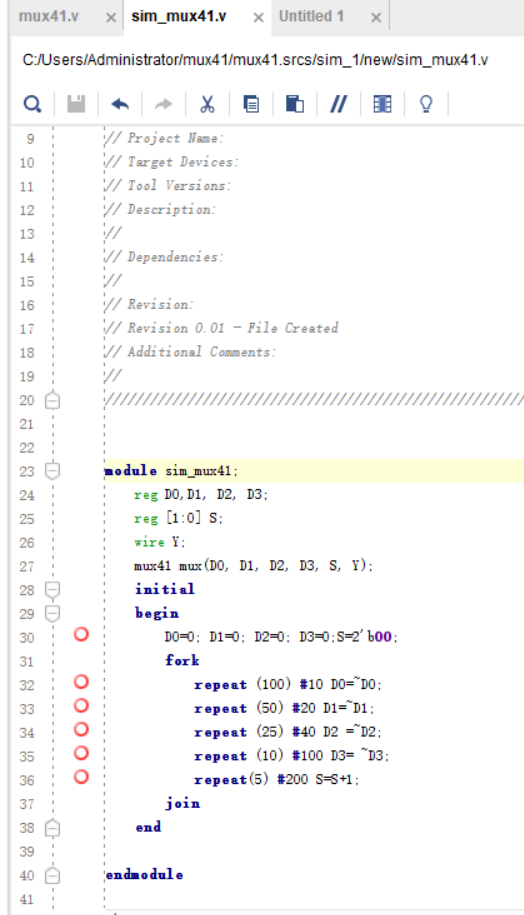
将两个与门的输出连接到一个或门中，作为输入信号。

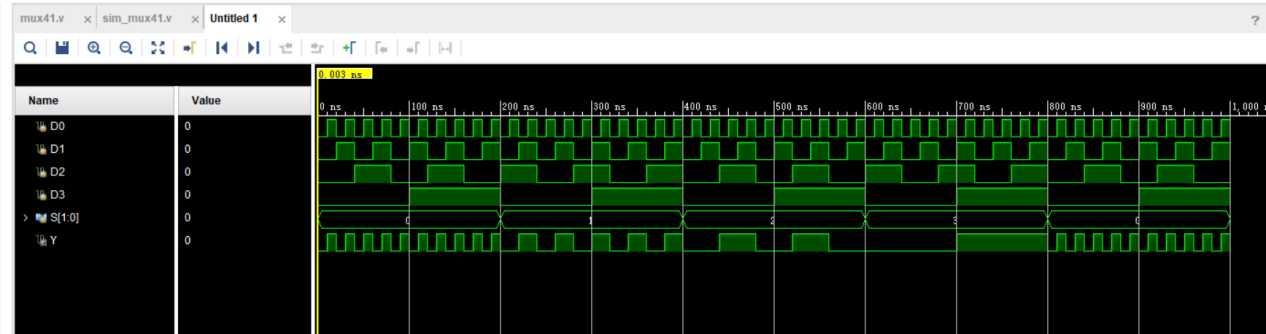
最后，将或门的输出作为Y的输出信号。

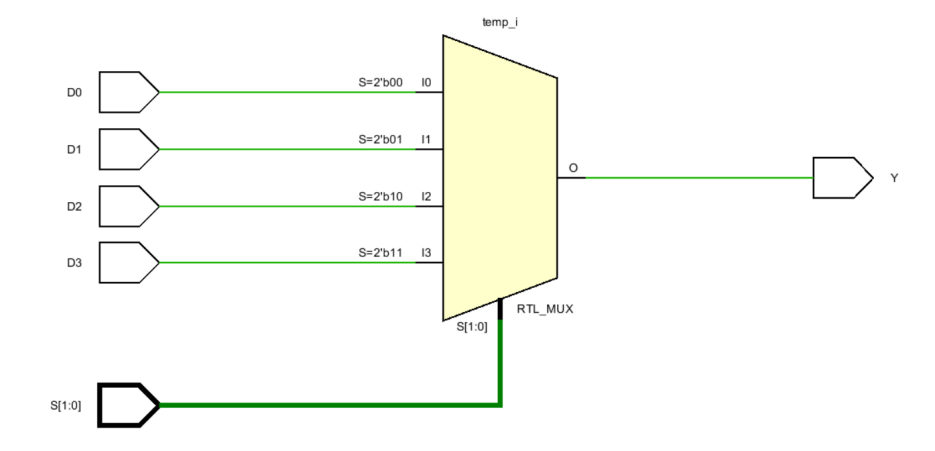
这样，当输入D0为逻辑0且S为逻辑0时，由于~S为逻辑1，与门1输出为逻辑0；当输入D1为逻辑0且S为逻辑1时，与门2输出为逻辑0；当输入D0为逻辑1且S为逻辑0时，与门1输出为逻辑1；当输入D1为逻辑1且S为逻辑1时，与门2输出为逻辑1。最终，或门将两个与门的输出进行逻辑或操作，也就是Y = D0⋅~S + D1⋅S的结果。

也可以直接通过表达式来构建电路；

Vivado:







有 4 个数据输入端口 D0、D1、D2 和 D3，2 个选择输入端口 S[1] 和 S[0]，以及 1 个数据输出端口 Y。

在 always 块中，使用 case 语句根据输入的选择信号 S 来选择相应的数据输入端口，将其赋值给临时变量 temp。如果选择信号 S 不在 00、01、10、11 这 4 个值中，则默认选择 D0。然后，使用 assign 语句将临时变量 temp 赋值给输出端口 Y。

在模块 sim\_mux41 中，我们定义了 4 个输入信号 D0、D1、D2 和 D3，以及 2 位选择信号 S，以及一个输出信号 Y。在 initial 块中，我们对输入信号进行了一些简单的测试。

首先，我们将 D0、D1、D2 和 D3 初始化为 0，并将选择信号 S 初始化为 00。然后，我们使用 fork-join 结构并行地执行了几个 repeat 语句，来改变输入信号的值。

repeat (100) #10 D0=~D0;：每 10 个时间单位，反转 D0 的值一次。这意味着 D0 的值会在 100 个时间单位内反转 10 次。

repeat (50) #20 D1=~D1;：每 20 个时间单位，反转 D1 的值一次。这意味着 D1 的值会在 50 个时间单位内反转 2.5 次，即反转 2 次。

repeat (25) #40 D2 =~D2;：每 40 个时间单位，反转 D2 的值一次。这意味着 D2 的值会在 25 个时间单位内反转 0.625 次，即反转 0 次。

repeat (10) #100 D3= ~D3;：每 100 个时间单位，反转 D3 的值一次。这意味着 D3 的值会在 10 个时间单位内反转 1 次。

此外，我们还使用了 repeat(5) #200 S=S+1; 语句，每 200 个时间单位，选择信号 S 的值加 1。这意味着选择信号 S 的值会在 5 个时间单位内从 00 变为 01，然后变为 10，最后变为 11。

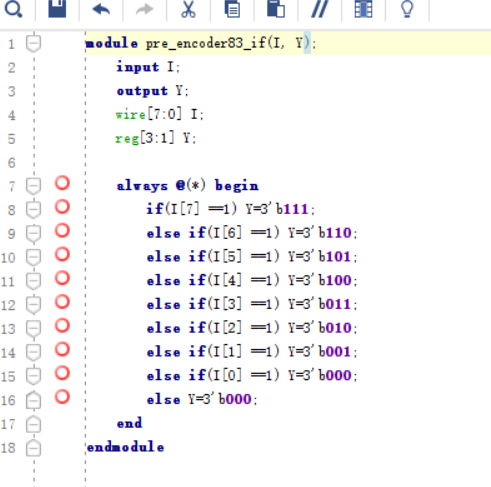
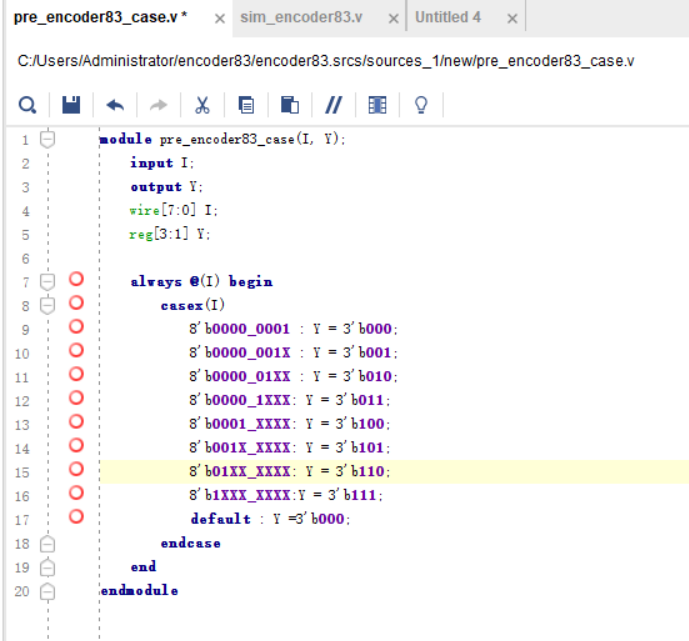
根据以上的输入变化，我们可以观察到输出信号 Y 的变化。

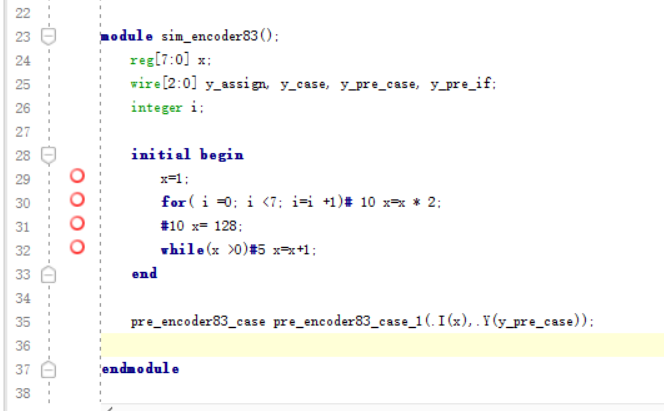
根据 mux41 模块的实现，当选择信号 S 的值为 00 时，输出信号 Y 的值等于输入信号 D0 的值；当选择信号 S 的值为 01 时，输出信号 Y 的值等于输入信号 D1 的值；当选择信号 S 的值为 10 时，输出信号 Y 的值等于输入信号 D2 的值；当选择信号 S 的值为 11 时，输出信号 Y 的值等于输入信号 D3 的值。如果选择信号 S 不在上述 4 个值中，则输出信号 Y 的值等于输入信号 D0 的值。

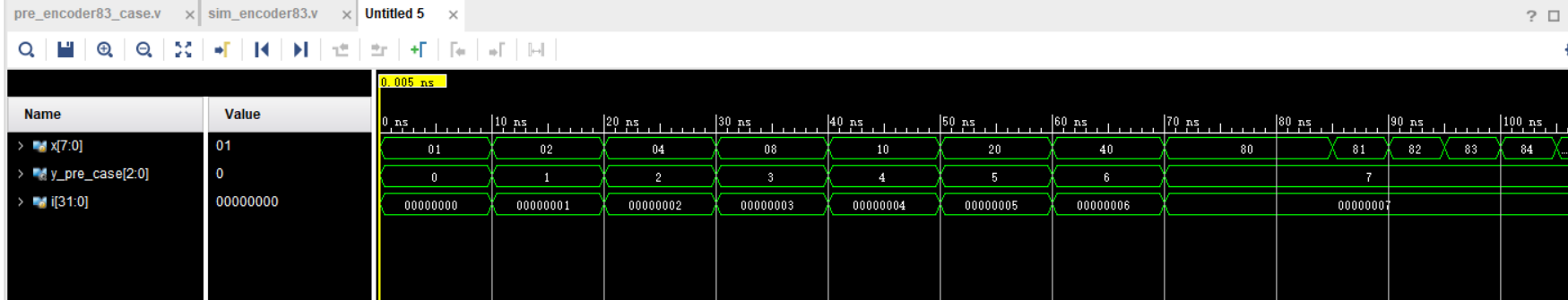
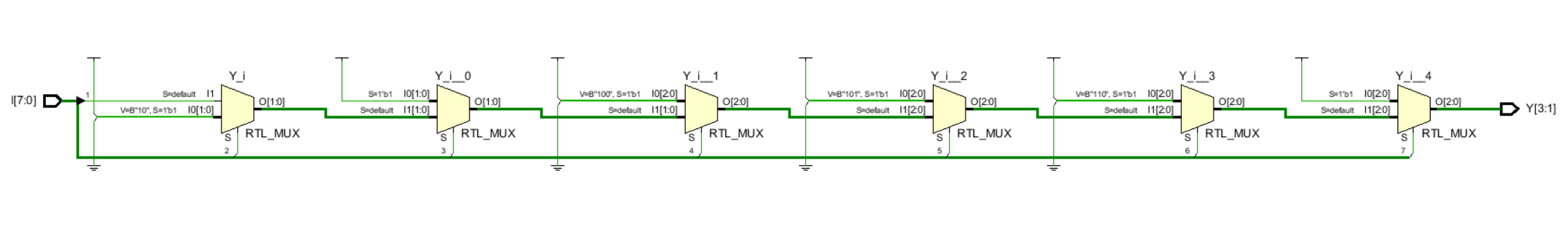
根据以上的输入变化和选择信号的变化，我们可以计算出对应的输出信号 Y 的值。在每个时间单位结束时，我们可以记录下输入信号 D0、D1、D2、D3 和选择信号 S 的值，以及输出信号 Y 的值。

编码器：

8-3优先编码器：







第一份程序的模块是一个3位预编码器的模块。它有一个8位的输入I和一个3位的输出Y。I被定义为一个8位的wire类型，而Y被定义为一个3位的reg类型。

在always块中，使用casex语句对输入I进行匹配。casex语句是一种在Verilog中用于多路选择的语句。它根据输入I的值来选择相应的输出Y的值。

casex语句中的每个匹配模式都是一个8位的二进制值，其中X表示可以是0或1的任意值。匹配模式的顺序是从上到下的，所以第一个匹配到的模式将被执行，而其他的模式将被忽略。

根据匹配模式，给输出Y赋予不同的值。例如，如果输入I的值为8'b0000\_0001，则输出Y的值将被赋为3'b000。如果输入I的值为8'b0000\_001X，则输出Y的值将被赋为3'b001。如果输入I的值不匹配任何模式，则执行default语句，将输出Y的值赋为3'b000。

这个预编码器的作用是将输入I的8位值编码成一个3位的预编码值，用于后续的处理或传输。根据不同的输入，它将输出不同的预编码值。

第二份的模块也是一个3位预编码器的模块。它有一个8位的输入I和一个3位的输出Y。I被定义为一个8位的wire类型，而Y被定义为一个3位的reg类型。

在always块中，使用if语句对输入I的每个位进行判断。if语句根据输入I的每个位的值来选择相应的输出Y的值。

首先，判断输入I的最高位I[7]的值是否为1。如果是1，则输出Y的值被赋为3'b111。如果不是1，则继续判断下一个位。

按照相同的方式，依次判断I的每个位的值。如果某个位的值为1，则输出Y的值将被赋为相应的3位预编码值，如3'b110、3'b101等。如果某个位的值不为1，则继续判断下一个位。如果所有位的值都不为1，则执行最后的else语句，将输出Y的值赋为3'b000。

sim\_encoder83是一个预编码器的测试模块。它实例化了一个待测的预编码器模块，并为其提供了输入信号x和输出信号y\_pre\_case。

在initial块中，首先给输入信号x赋初值为1。然后使用一个for循环，每延迟10ns，将x信号左移一位，共执行7次。这样可以模拟输入信号x的变化。接着，延迟10ns后，将x信号赋值为128，这样可以模拟输入信号x的一个特定值。然后使用一个while循环，每延迟5ns，将x信号减1，直到x信号的值为0。这样可以模拟输入信号x的递减过程。

在测试模块中，实例化了一个待测的预编码器模块pre\_encoder83\_case，并将输入信号x连接到该模块的输入端口I，将输出信号y\_pre\_case连接到该模块的输出端口Y。通过这样的实例化，可以对待测的预编码器模块进行功能验证。

它可以对预编码器模块进行仿真测试，验证其在不同输入信号x下的输出信号y\_pre\_case是否符合预期的预编码规则。

译码器：

24译码器：

图形用户界面, 文本, 应用程序, Word

描述已自动生成日历

描述已自动生成

图示

描述已自动生成

将两个输入线分别标记为A1和A0。

使用一个非门将A1进行反相得到~A1。

使用一个非门将A0进行反相得到~A0。

使用一个与门将A1和~A0作为输入，并将其输出连接到输出线Y2上。

使用一个与门将A1和A0作为输入，并将其输出连接到输出线Y3上。

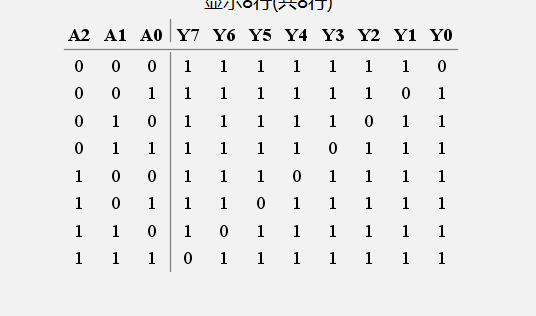
使用一个与门将~A1和A0作为输入，并将其输出连接到输出线Y1上。

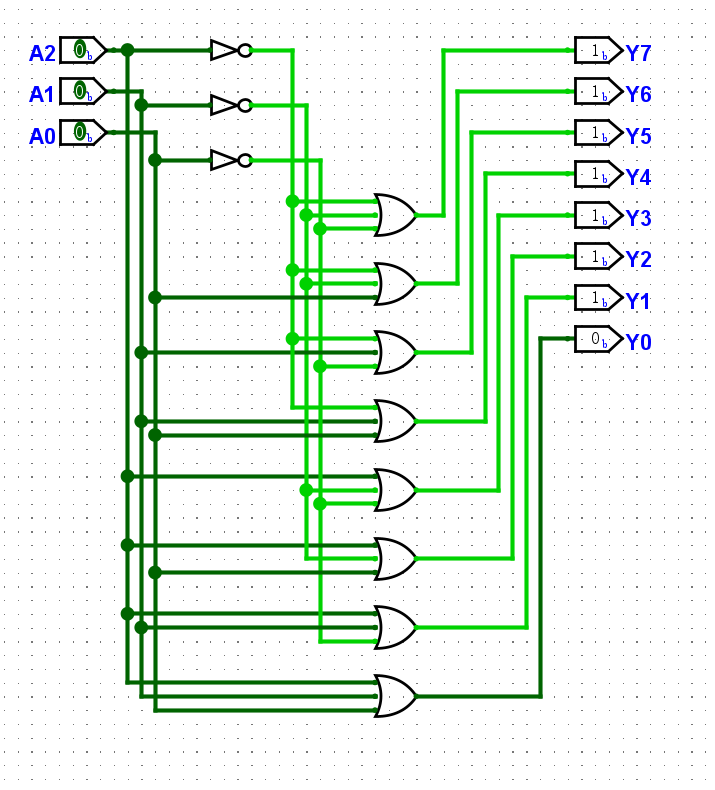
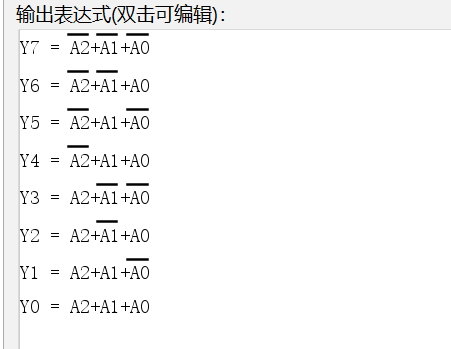
使用一个与门将~A1和~A0作为输入，并将其输出连接到输出线Y0上。

实现了简易的2-4译码器的功能

也可以直接通过四个表达式来构建电路，便可以直接得到上图所示电路；

38译码器：





38译码器是一种数字电路，用于将3位二进制输入转换成8位二进制输出。输入和输出之间的关系是一一对应的，即每一种输入组合都对应着唯一的输出值。具体的关系如上真值表。

输入A2A1A0三位二进制位，将其转化为对应十进制位输出0，其他都输出1.Y7为1时需要000才行，借此我们可以轻易写出表达式三个取反相加即可。最后根据表达式构建电路图如上所示。

1. **调试和心得体会**

首先，我使用logisim实现了多输入门电路。通过这个实验，我深入理解了多输入门的工作原理，以及如何使用逻辑门实现复杂的逻辑功能。

接下来，我使用logisim和vivado实现了复用器即多路选择器。通过这个实验，学会了如何使用复用器将多个输入信号选择输出，并且实现了多种复杂的逻辑功能。

然后，我使用vivado实现了编码器和83优先编码器。通过这个实验，我学习了编码器的工作原理以及如何将多个输入信号编码成少量的输出信号。

最后，我使用logisim实现了24译码器和38译码器。通过这个实验，我学习了译码器的工作原理以及如何将少量的输入信号解码成多个输出信号。在调试过程中，我们需要仔细检查输入信号和输出信号的正确连接，以及输出信号的正确性。

通过以上实验，我们不仅掌握了多个重要电路的实现方法，还学会了使用logisim和vivado进行电路的调试和仿真。在实验过程中，我们遇到了一些问题，但通过仔细分析和调试，最终成功解决了这些问题。通过实验，我们深入理解了计算机组成原理中的重要概念和原理，提高了我们的实践能力和解决问题的能力。

本次实验复习了上学期数电实验的内容，重新对vivado编程语言有了新的认识，并且还学习了logisim软件来实现一些逻辑电路。

总的来说，计算机组成原理的实验课让我们在实践中学习了多个重要的电路的实现和调试方法，提高了我们的实践能力和解决问题的能力。通过实验，我们更加深入地理解了计算机组成原理中的重要概念和原理，为我们进一步学习和研究计算机组成原理打下了坚实的基础。