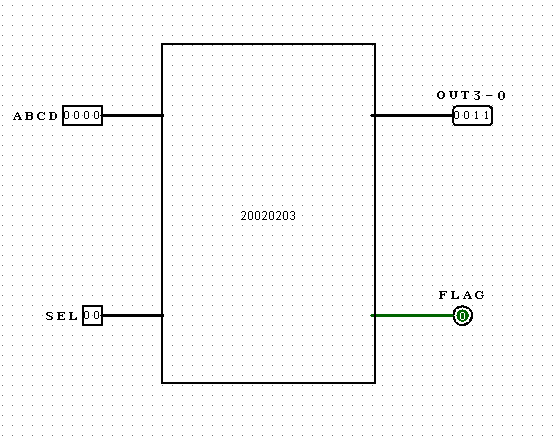
**ALU、译码器、编码器、立即数扩展的设计与综合验证**

1. **基于Logisim平台的电路测试**
   1. **应用测试截图**

****

* 1. **电路测试填写下表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **当SEL=00时** | **当SEL=01时** | **当SEL=10时** | **当SEL=11时** |  |
| **输入** | **输出** | **输出** | **输出** | **输出** | **输出** |
| **A B C D** | **OUT3-0** | **OUT3-0** | **OUT3-0** | **OUT3-0** | **FLAG** |
| **0 0 0 0** | **0011** | **0000** | **0000** | **0000** | **0** |
| **0 0 0 1** | **0100** | **0001** | **0001** | **0000** | **0** |
| **0 0 1 0** | **0101** | **0010** | **0011** | **0000** | **0** |
| **0 0 1 1** | **0110** | **0011** | **0010** | **0000** | **0** |
| **0 1 0 0** | **0111** | **0100** | **0110** | **0000** | **0** |
| **0 1 0 1** | **1000** | **1011** | **0111** | **0000** | **0** |
| **0 1 1 0** | **1001** | **1100** | **0101** | **0000** | **0** |
| **0 1 1 1** | **1010** | **1101** | **0100** | **0000** | **0** |
| **1 0 0 0** | **1011** | **1110** | **1100** | **0000** | **0** |
| **1 0 0 1** | **1100** | **1111** | **1000** | **0000** | **0** |
| **1 0 1 0** | **0000** | **0000** | **0000** | **0000** | **1** |
| **1 0 1 1** | **0000** | **0000** | **0000** | **0000** | **1** |
| **1 1 0 0** | **0000** | **0000** | **0000** | **0000** | **1** |
| **1 1 0 1** | **0000** | **0000** | **0000** | **0000** | **1** |
| **1 1 1 0** | **0000** | **0000** | **0000** | **0000** | **1** |
| **1 1 1 1** | **0000** | **0000** | **0000** | **0000** | **1** |

* 1. **分析逻辑功能**

(1)SEL=00时，将输入的自然二进制码转换为余3码输出；并且转换功能生效时FLAG输出0，否则输出1;

(2)SEL=01时，将输入的自然二进制码转换为2421码输出；并且转换功能生效时FLAG输出0，否则输出1;

(3)SEL=10时，将输入的自然二进制码转换为格雷码（循环码）输出；并且转换功能生效时FLAG输出0，否则输出1;

(4)SEL=11时，对输入的自然二进制码进行检测，当表示的十进制数为0~9时，FLAG输出0，当为10~15时，输出1。

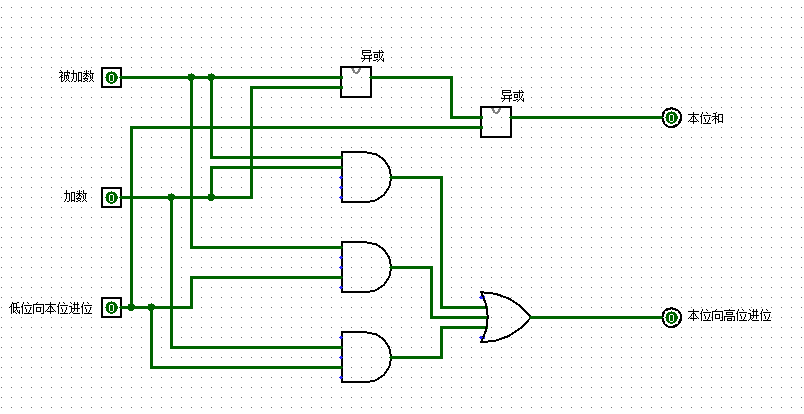
* 1. **理解**

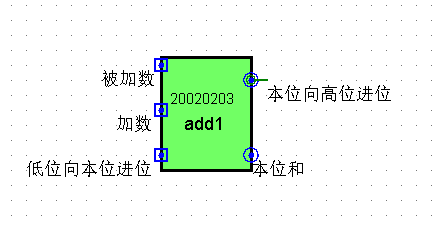
该电路可以实现多种类型的代码转换电路，首先分别构建了各自的代码转换电路，然后通过选择信号SEL进行控制输出，使用“数据选择器”模块，当SEL不同时，对应得到输出信号也就不同，实现了多种类型代码转换的功能。

二、**设计能完成四种运算的32位ALU**

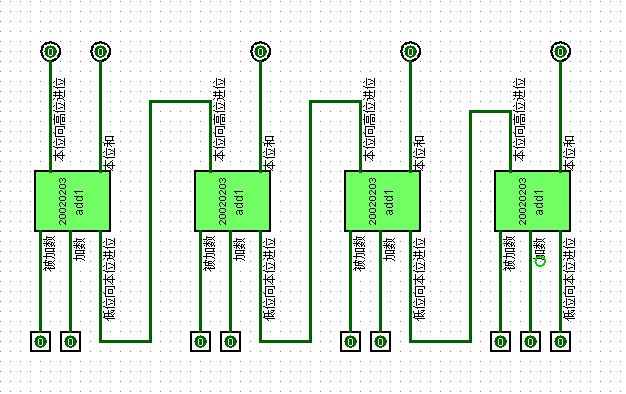
**1、全加器**

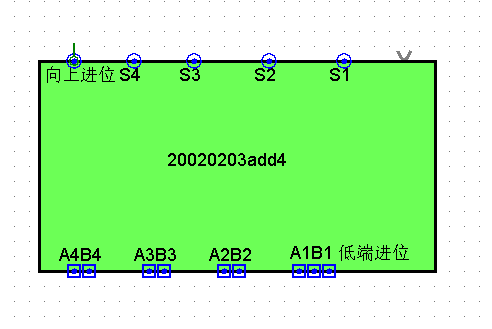
**（1）1位全加器**



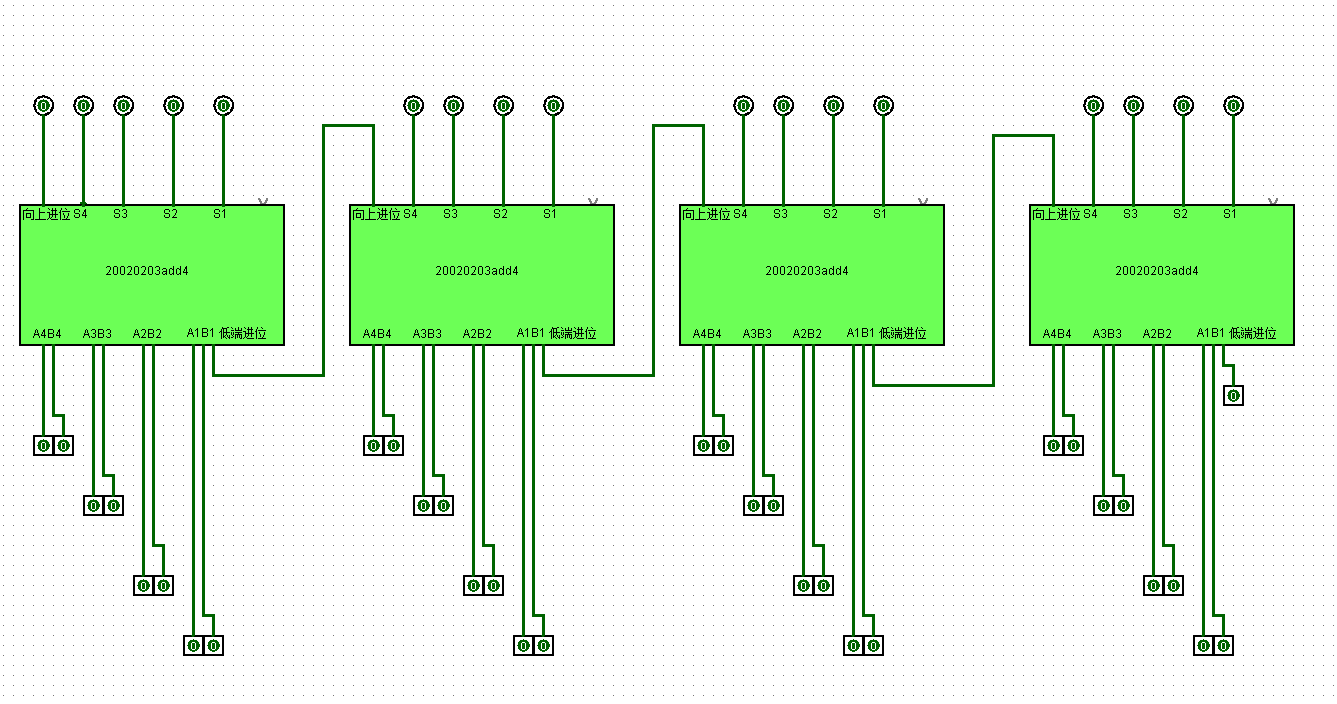


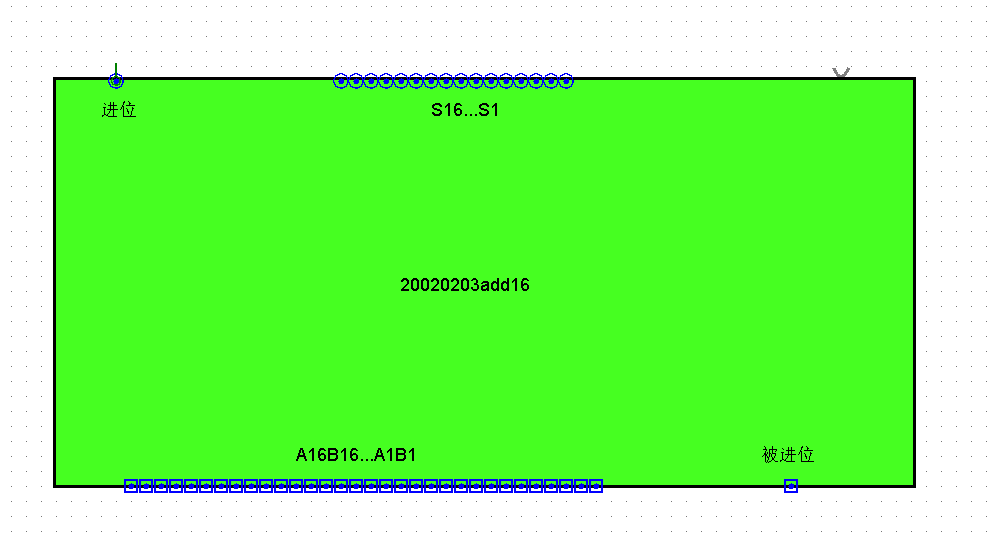
**（2）4位全加器**



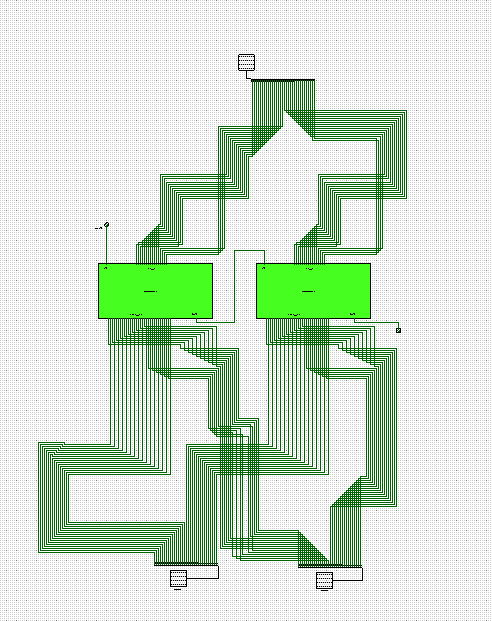


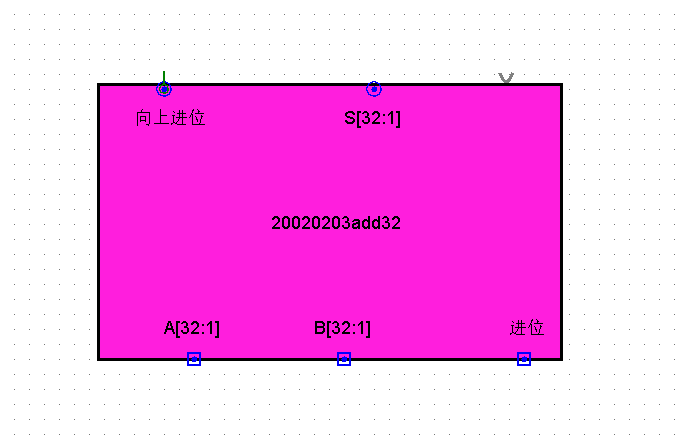
**（3）16位全加器**



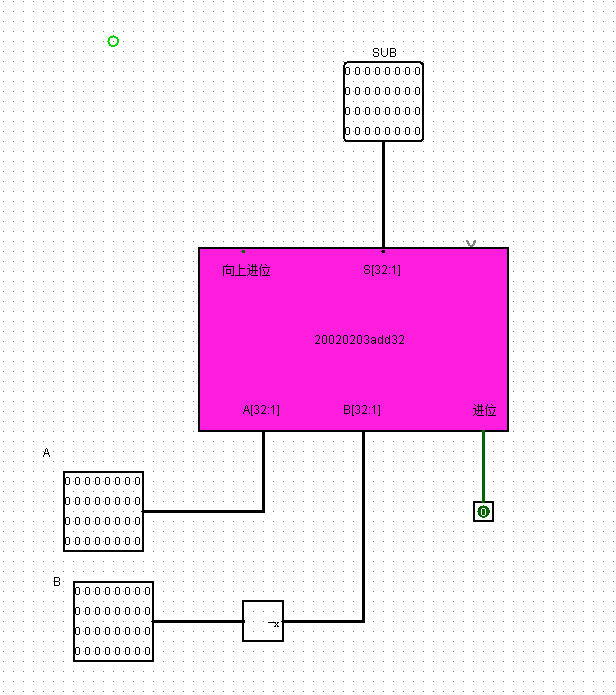


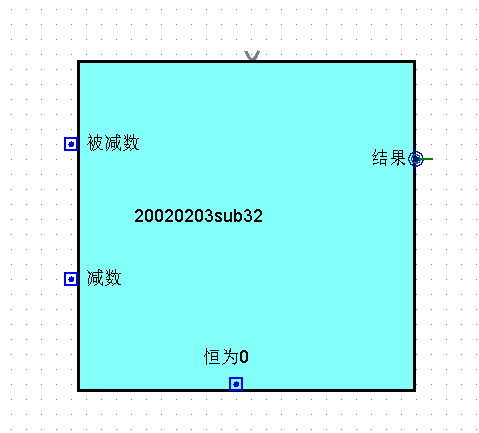
**（4）32位全加器**



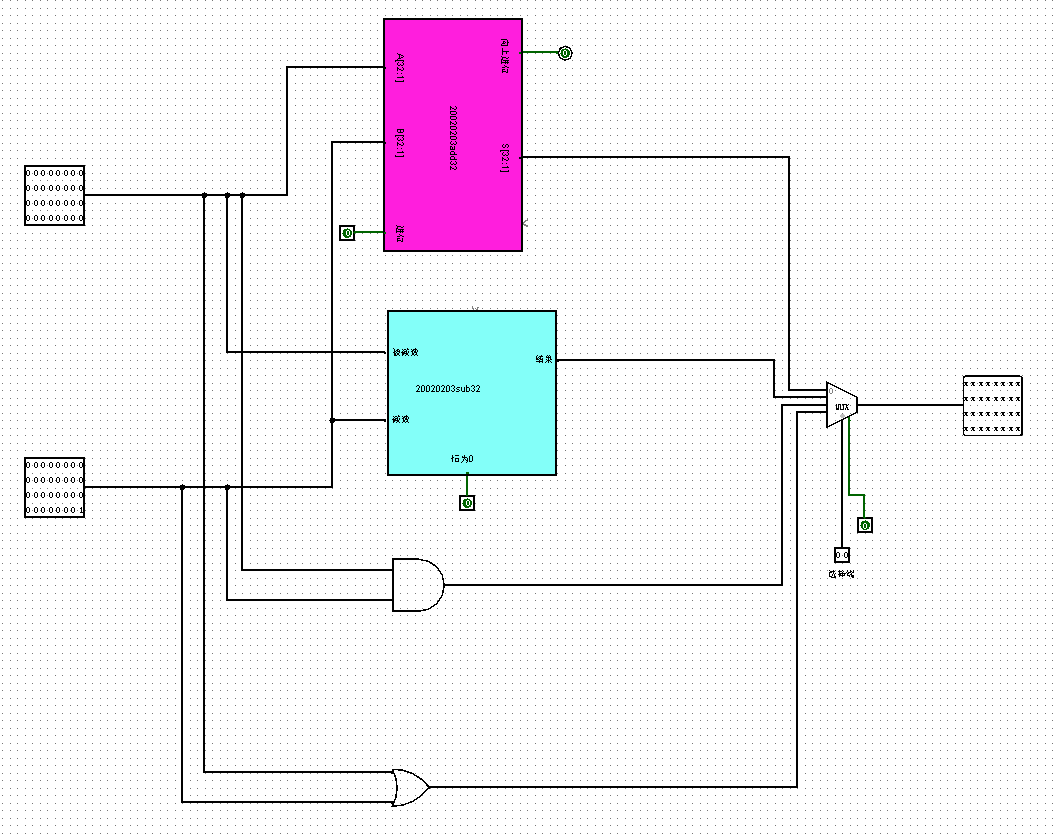


**2、32位全减器**

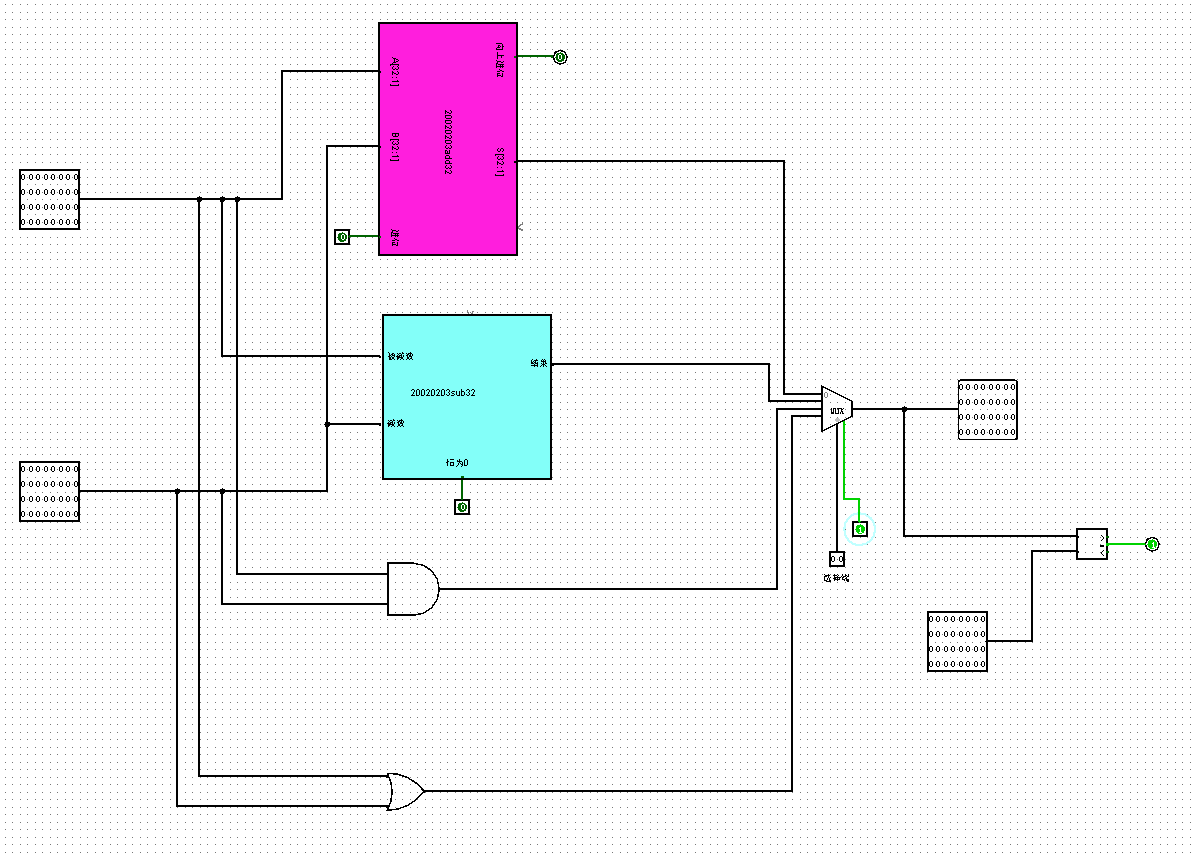




1. **能完成四种运算的ALU**

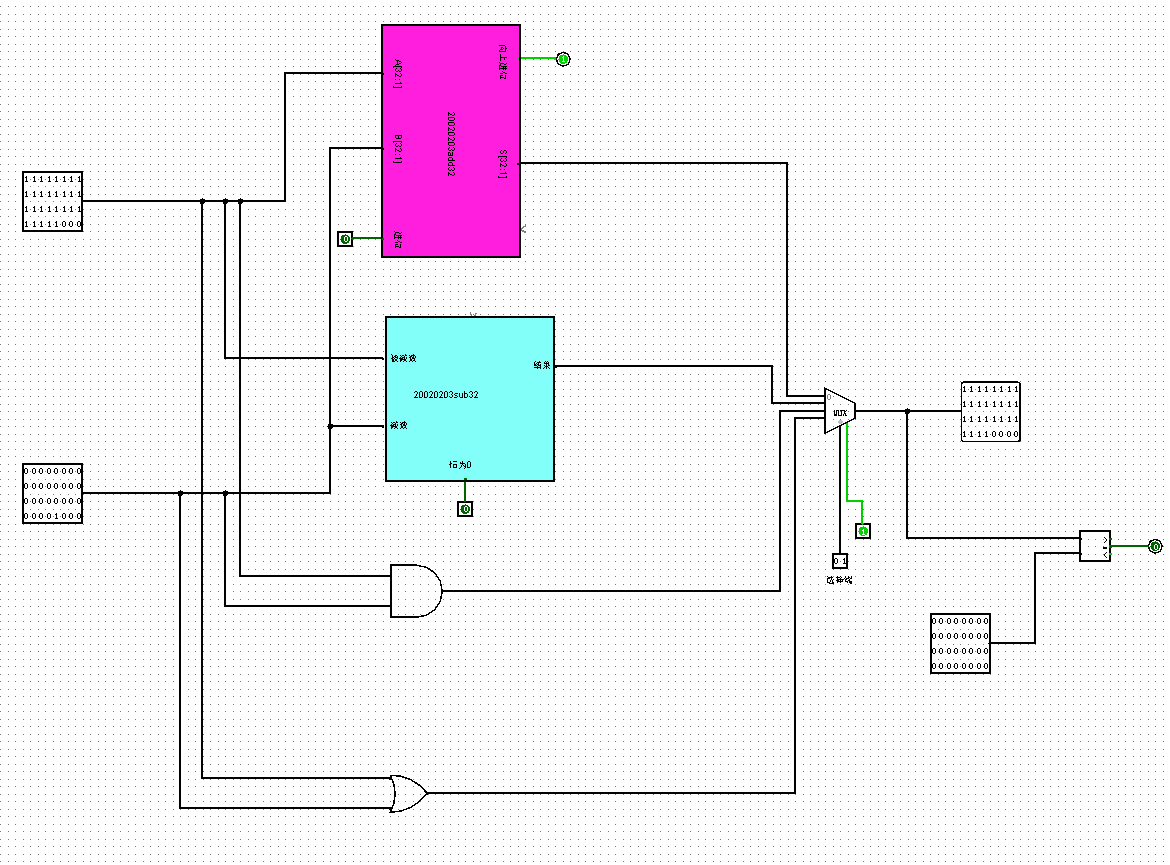
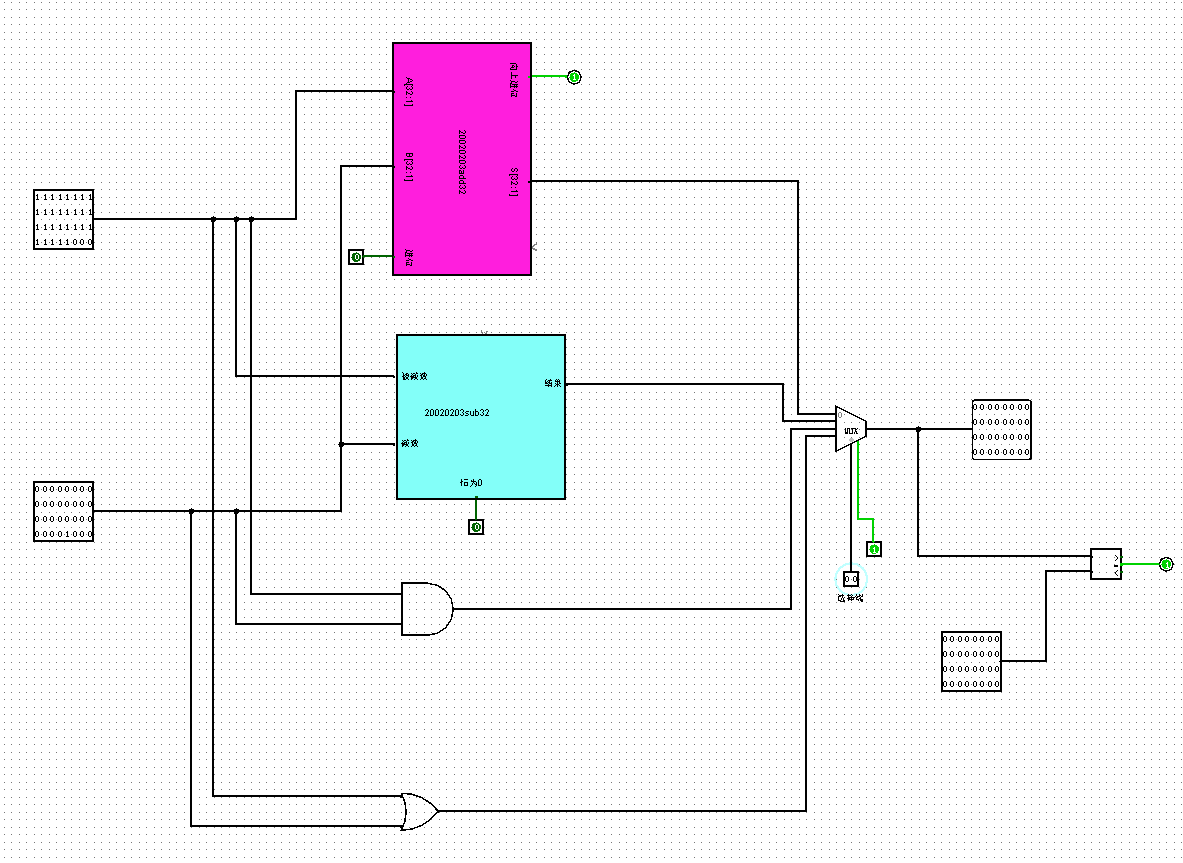


**4、添加运算结果为零的标志信号zero**



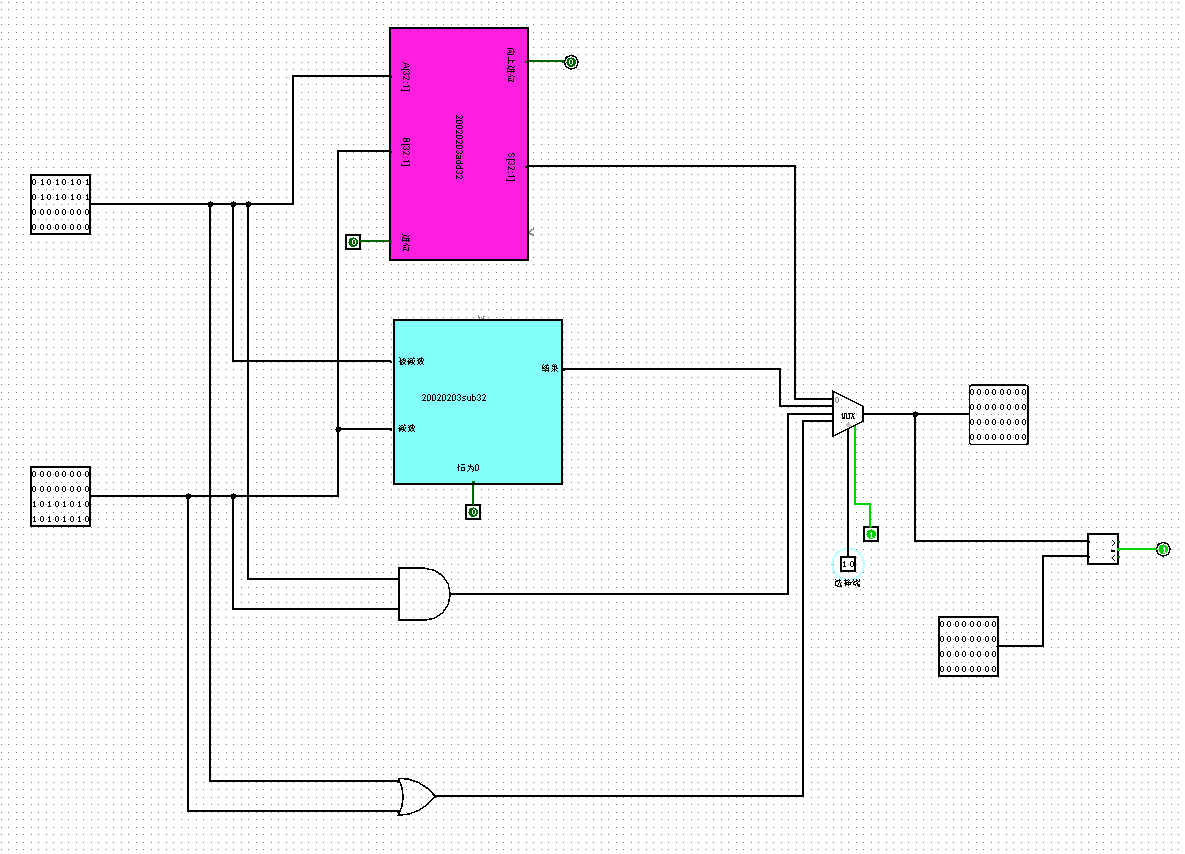
**5．算术运算（补码）测试：若(A)10 = -8，(B)10 = 8，**

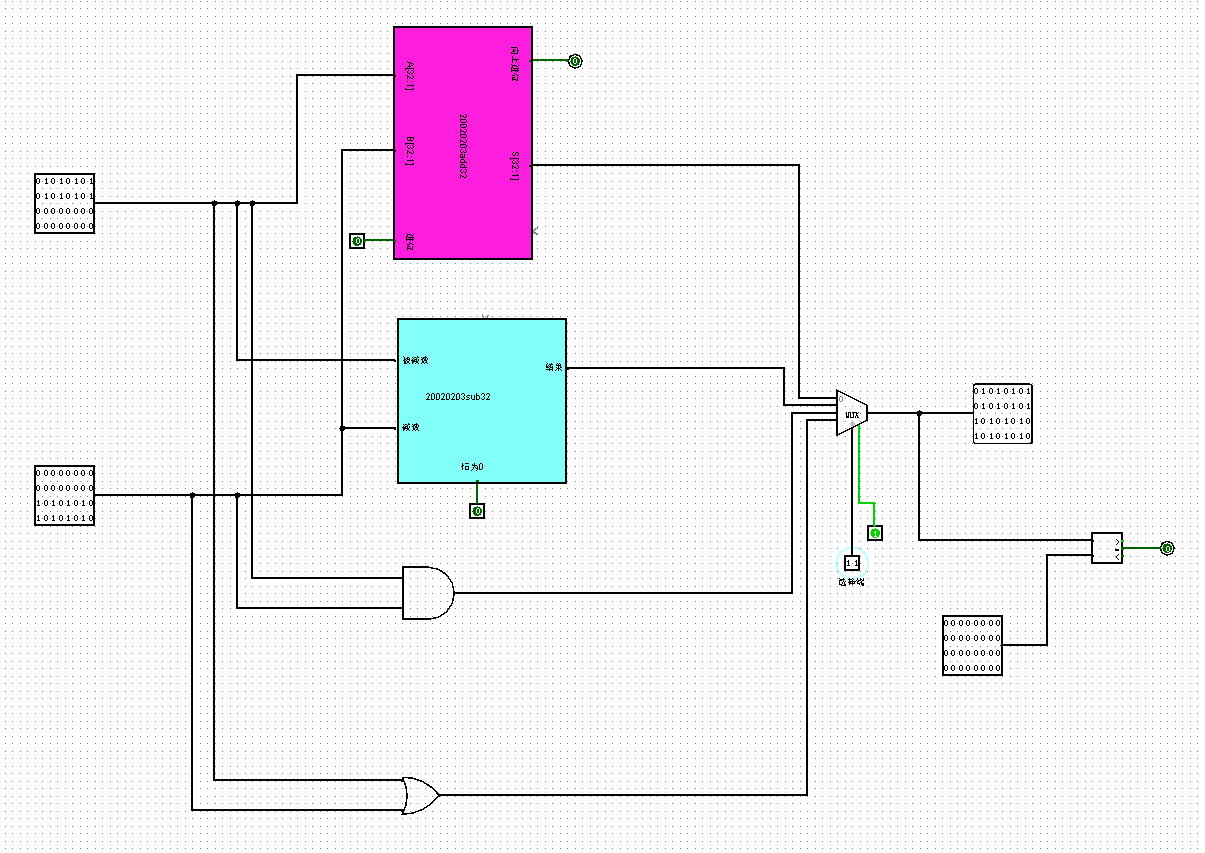
|  |  |  |  |
| --- | --- | --- | --- |
| **ALU\_sel** | **ALU\_out** | **（ALU\_out）10** | **zero** |
| **00** | **C:\Users\user\AppData\Local\Temp\1638023537(1).png** | **0** | **C:\Users\user\AppData\Local\Temp\1638023568(1).png** |
| **01** | **C:\Users\user\AppData\Local\Temp\1638024113(1).png** | **-16** | **C:\Users\user\AppData\Local\Temp\1638024132(1).png** |



**6．逻辑运算测试：若(A)16=55550000，(B)16=0000AAAA，**

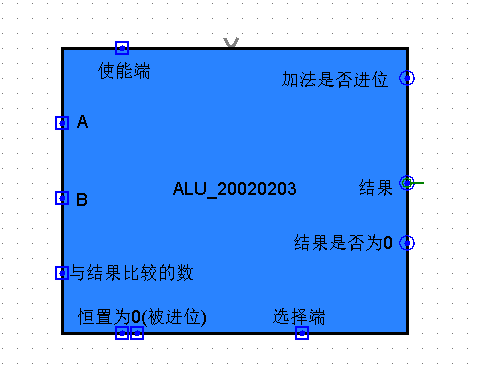
|  |  |
| --- | --- |
| **ALU\_sel** | **ALU\_out** |
| **10** | **C:\Users\user\AppData\Local\Temp\1638024434(1).png** |
| **11** | **C:\Users\user\AppData\Local\Temp\1638024451(1).png** |



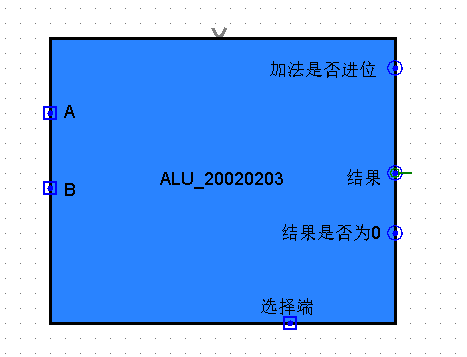


**7．编辑逻辑符号（ALU-学号），截图备用。**

**(1)更改前**



**(2)更改后(为部分地方赋予常量值，这样就可以减少一部分输入端)**

****

**8．将这个ALU的Verilog HDL描述补充完整（模块名含学号）。**

**module ALU\_20020203(A,B,ALU\_sel,ALU\_out,zero);**

**input [32:1] A,B;**

**input [1:0] ALU\_sel;**

**output [32:1] ALU\_out;**

**output zero;**

**reg zero;**

**assign zero=(ALU\_out==32’b0)?1:0;**

**always @(A or B or ALU\_sel)**

**case (ALU\_sel)**

**0:ALU\_out=A+B;**

**1: ALU\_out=A-B;**

**2: ALU\_out=A&B;**

**3: ALU\_out=A|B;**

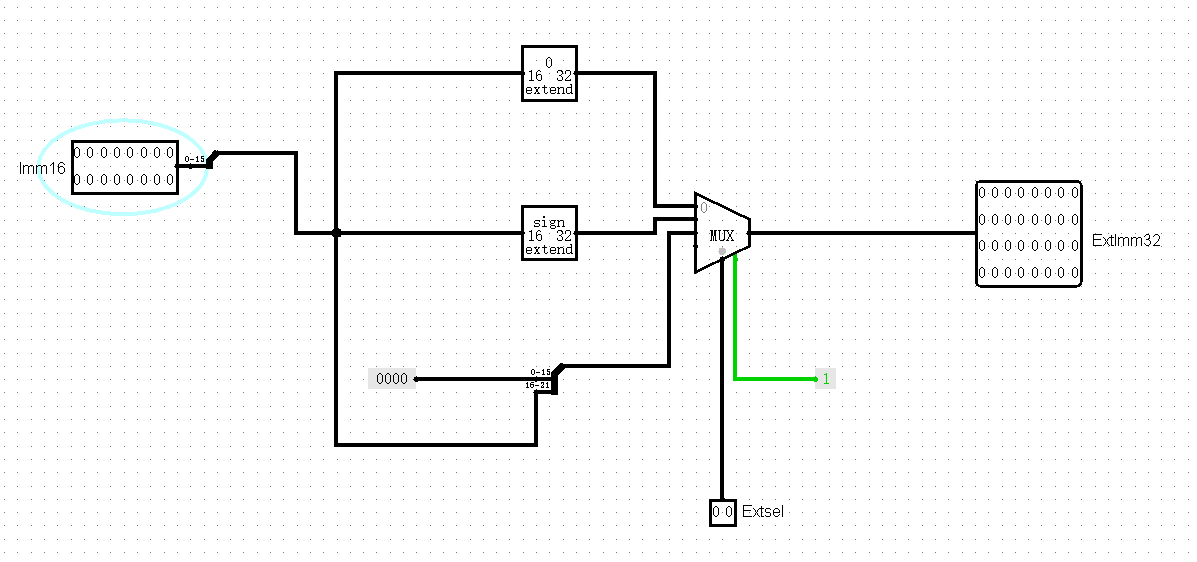
**default: ALU\_out=32’b0;**

**endcase**

**endmodule**

三、**立即数扩展电路的设计与验证**

**1、利用Logisim中的“分线器”和“选择器”，构造立即数扩展电路。**

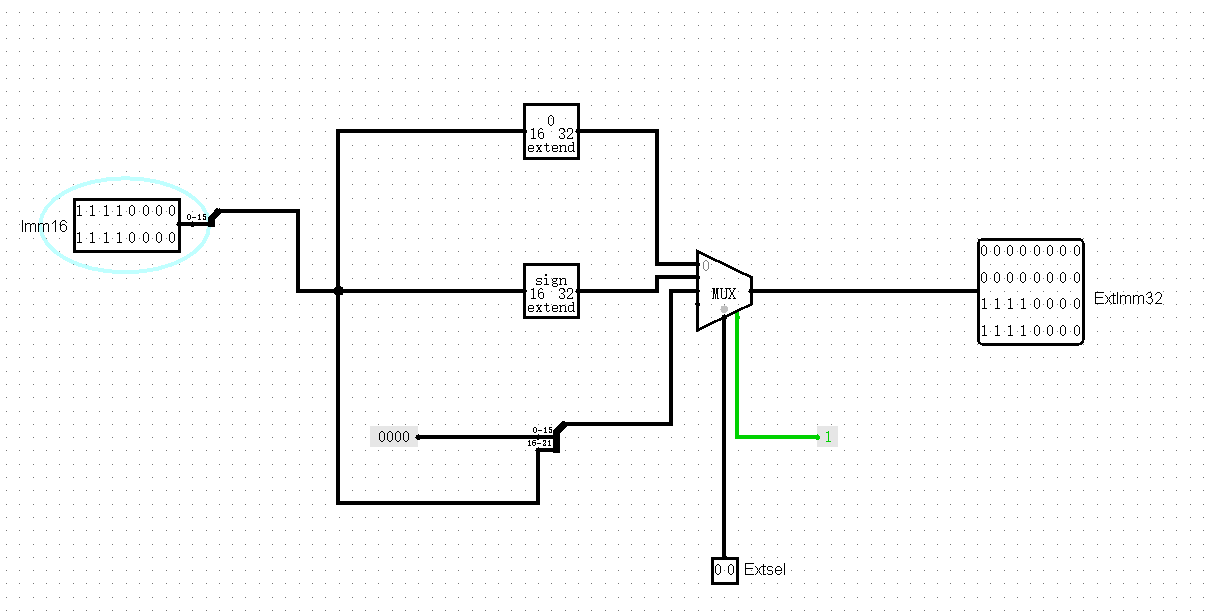
****

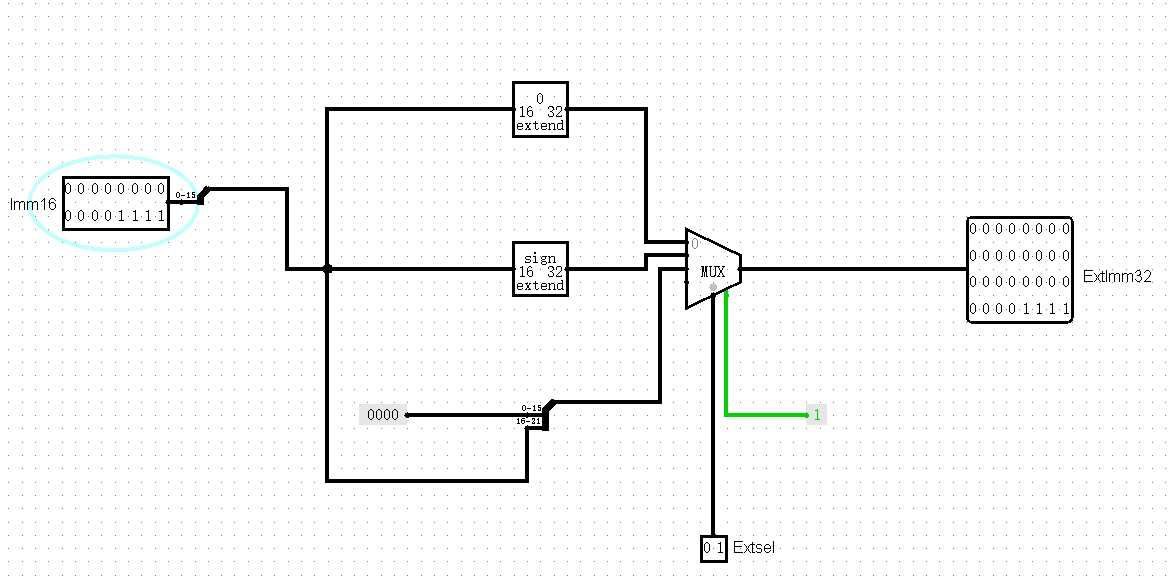
**2、功能表**

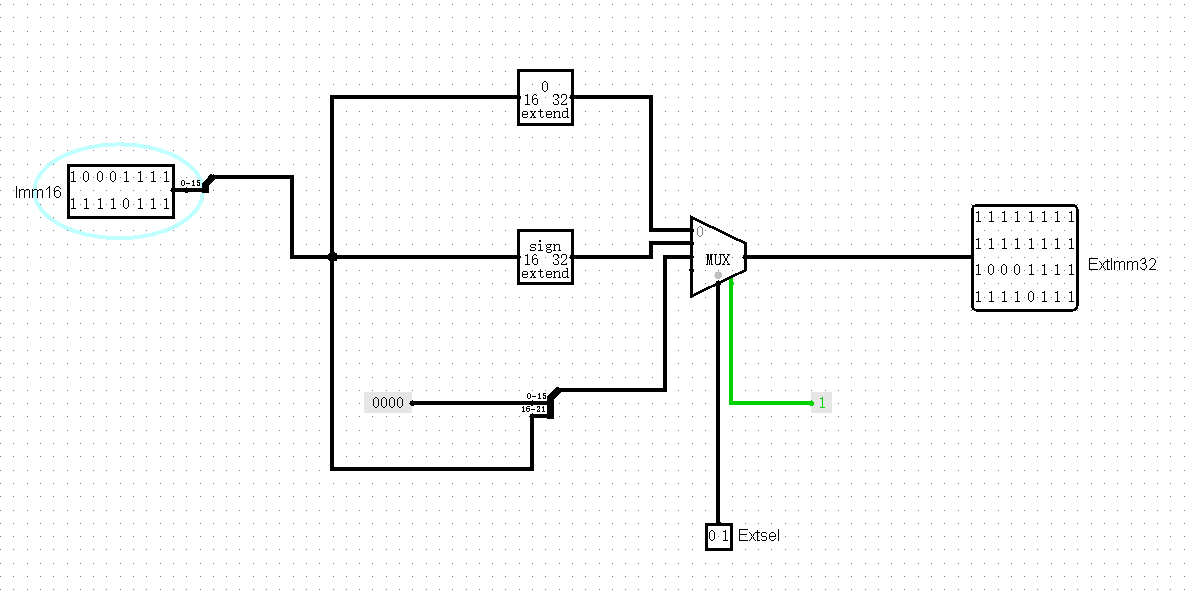
|  |  |  |  |
| --- | --- | --- | --- |
| **信号名** | **位宽** | **方向** | **说明** |
| **Imm16** | **16** | **输入** | **来自指令寄存器的16位立即数** |
| **Extsel** | **2** | **输入** | **00：无符号扩展，将16位立即数进行0扩展至32位立即数；**  **01：符号扩展，将16位补码立即数扩展成32位补码立即数；**  **10：低位0扩展，将16位立即数移至32位立即数的高16位，低16位补0。** |
| **ExtImm32** | **32** | **输出** | **扩展后的32位立即数** |

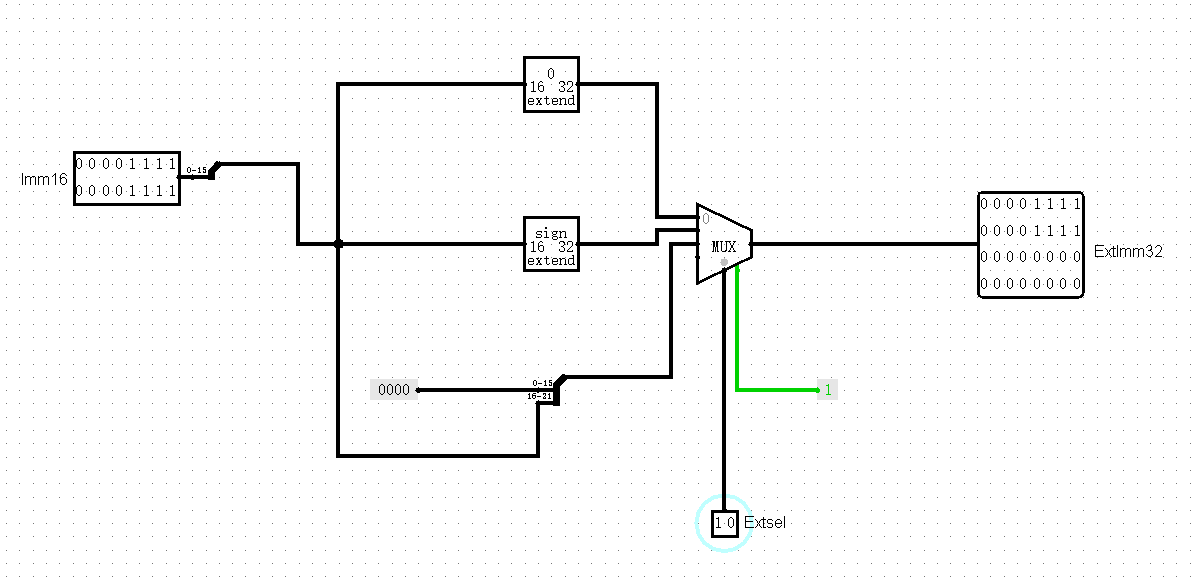
**3、模拟验证，填表下表**

|  |  |  |
| --- | --- | --- |
| **Imm16** | **Extsel** | **ExtImm32** |
| **1111000011110000** | **00** | **C:\Users\user\AppData\Local\Temp\1638030674(1).png** |
| **0000000000001111** | **01** | **C:\Users\user\AppData\Local\Temp\1638030726(1).png** |
| **1000111111110111** | **01** | **C:\Users\user\AppData\Local\Temp\1638030796(1).png** |
| **0000111100001111** | **10** | **C:\Users\user\AppData\Local\Temp\1638030865(1).png** |

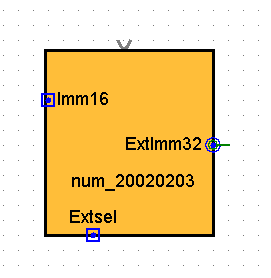








4、**编辑逻辑符号，备用。**

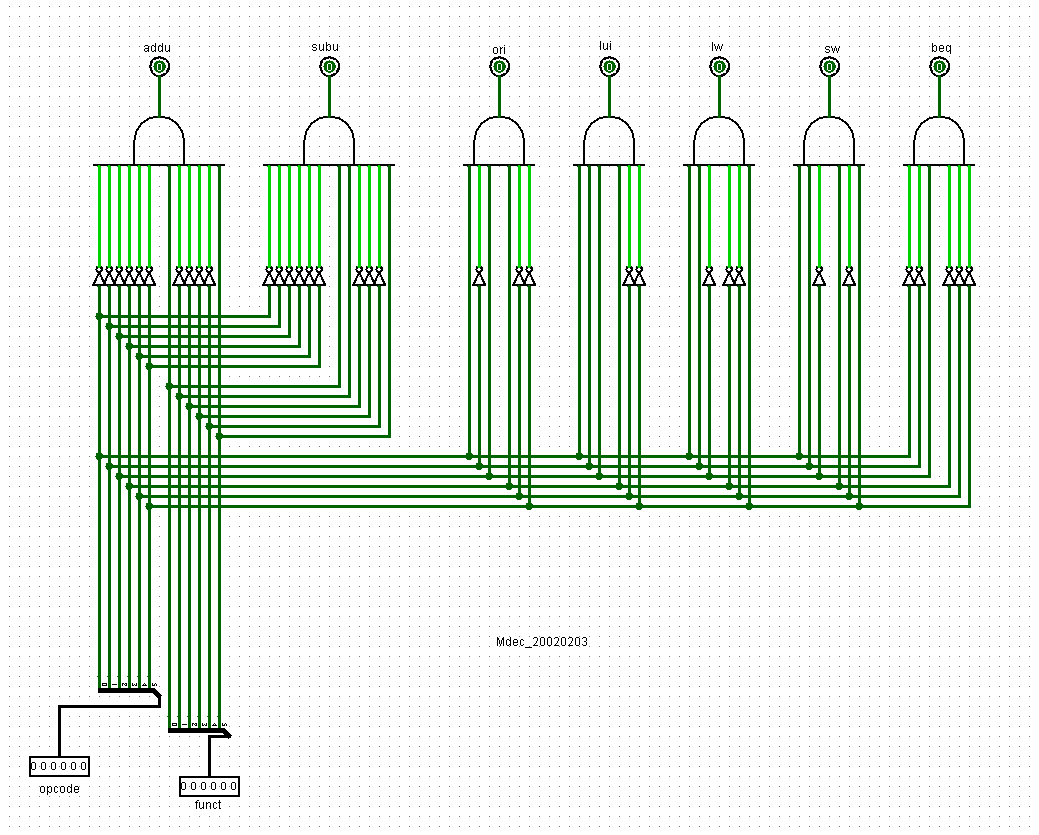


**四、7条MIPS指令的译码电路设计与验证**

**1．7条指令说明见相关ppt，根据7条MIPS指令的特征码“opcode”和“funct”，设计指令译码器。**

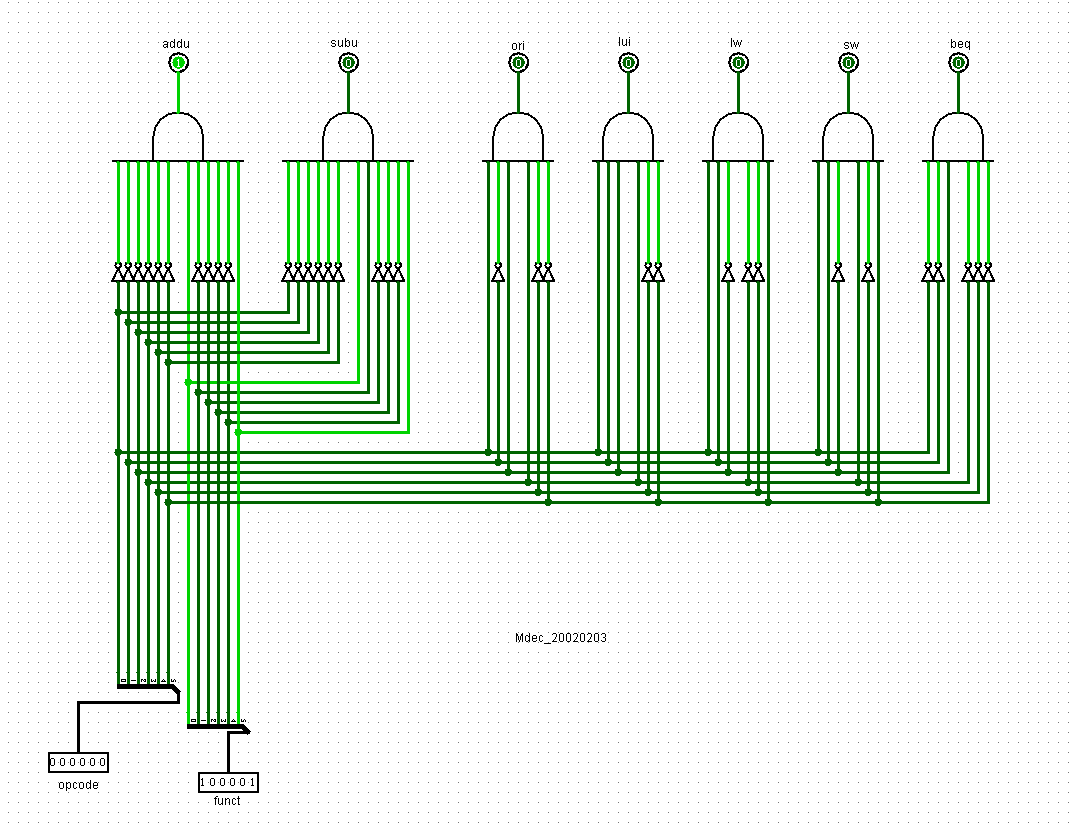
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **输入** | | **输出** | | | | | | |
|  |  | **加法** | **减法** | **或立即数** | **立即数置高位** | **取字** | **存字** | **相等跳转** |
| **opcode** | **funct** | **addu** | **subu** | **ori** | **lui** | **lw** | **sw** | **beq** |
| **000000** | **100001** | **1** | **0** | **0** | **0** | **0** | **0** | **0** |
| **000000** | **100011** | **0** | **1** | **0** | **0** | **0** | **0** | **0** |
| **001101** | **X** | **0** | **0** | **1** | **0** | **0** | **0** | **0** |
| **001111** | **X** | **0** | **0** | **0** | **1** | **0** | **0** | **0** |
| **100011** | **X** | **0** | **0** | **0** | **0** | **1** | **0** | **0** |
| **101011** | **X** | **0** | **0** | **0** | **0** | **0** | **1** | **0** |
| **000100** | **X** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |

**【设计实现】**

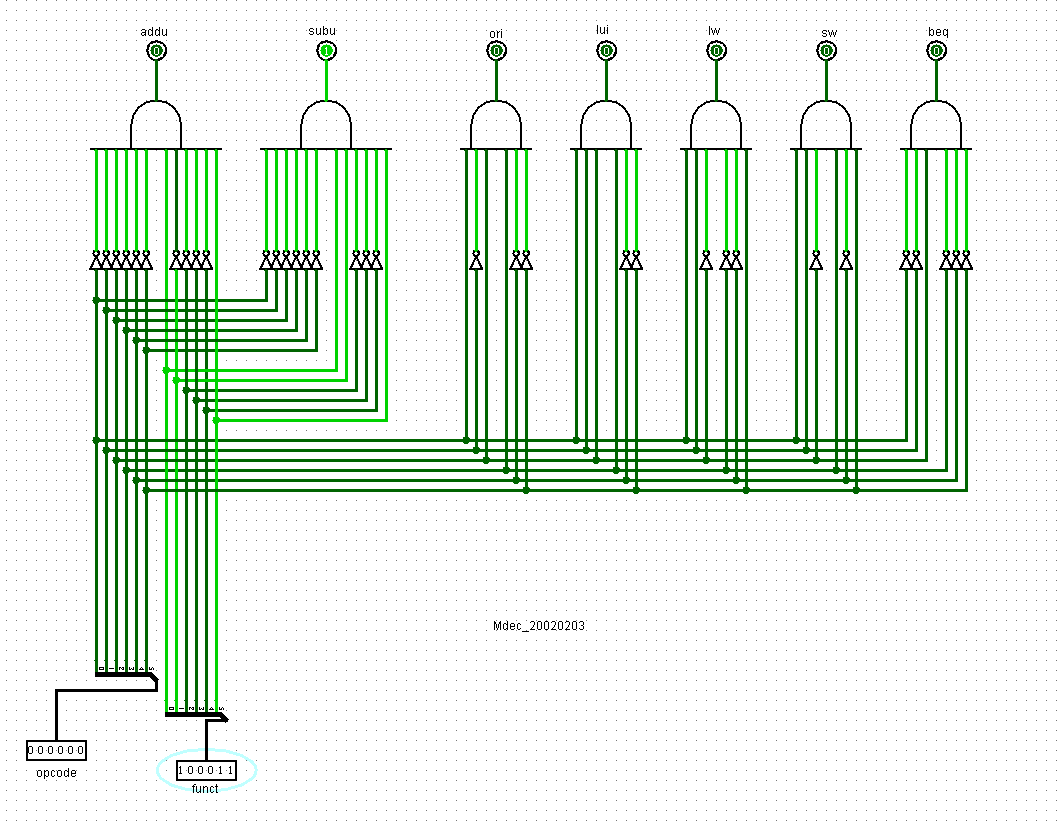


**2．在Logisim中设计实现指令译码器，进行验证，并截取正确结果电路图。**

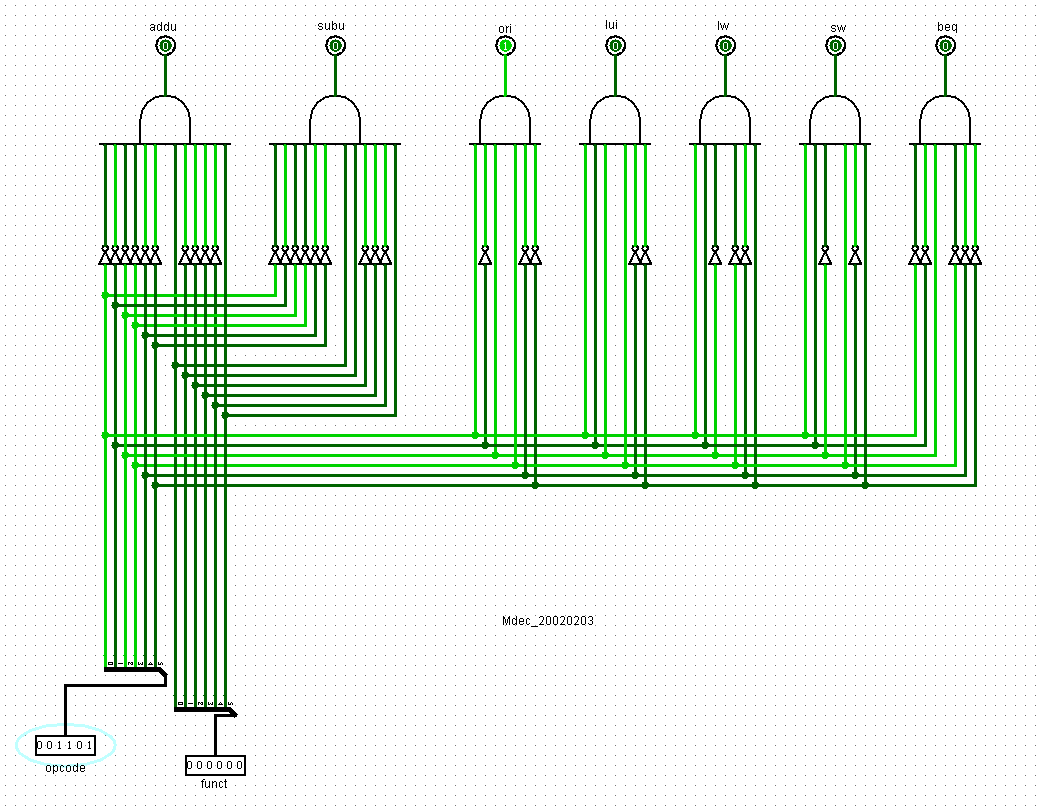
**【addu】**



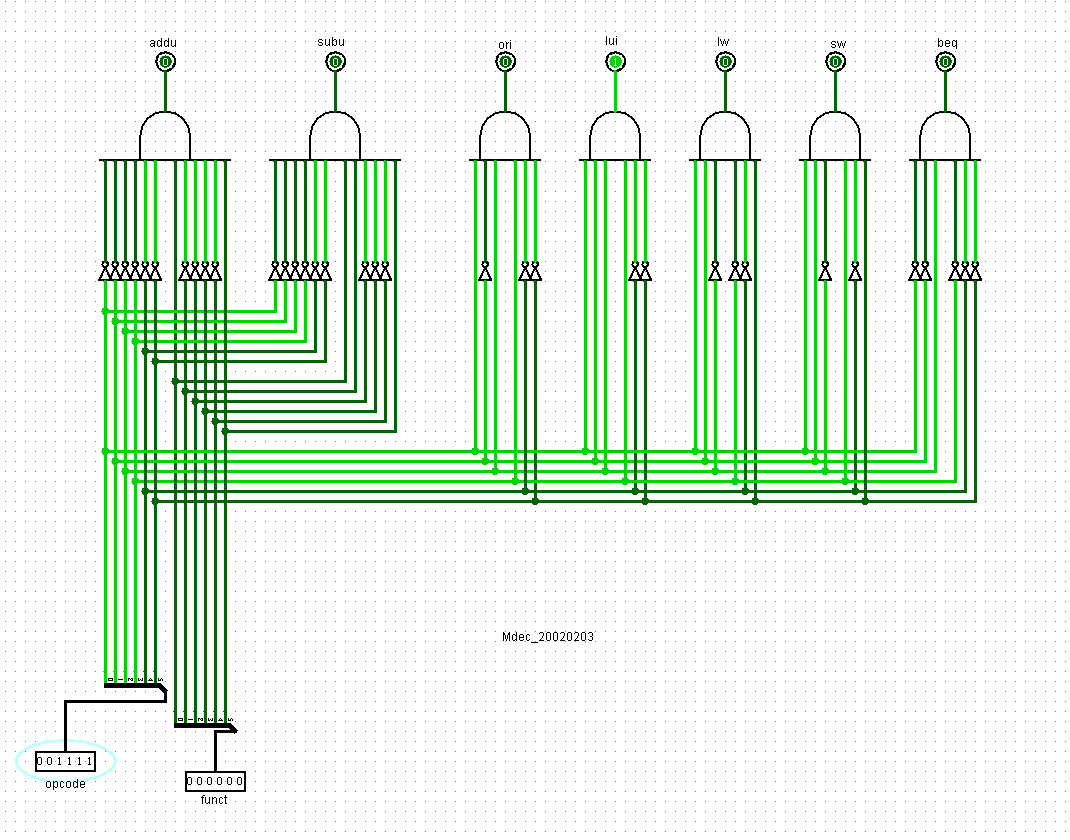
【subu】



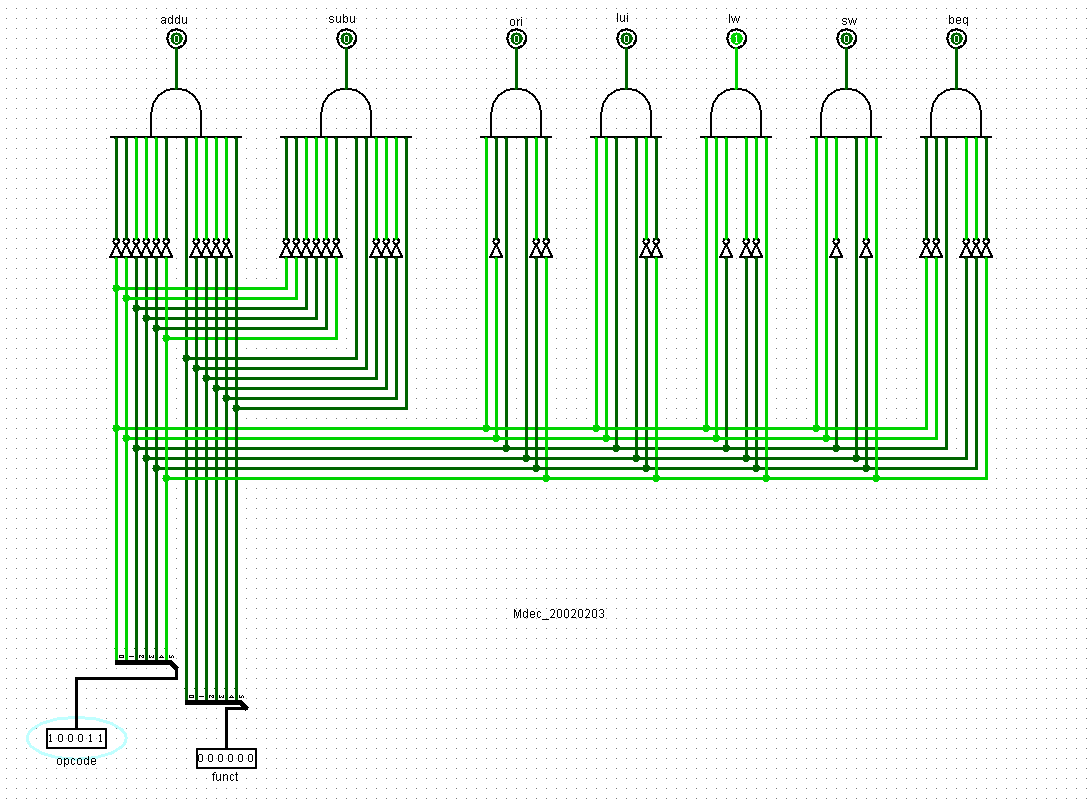
【ori】



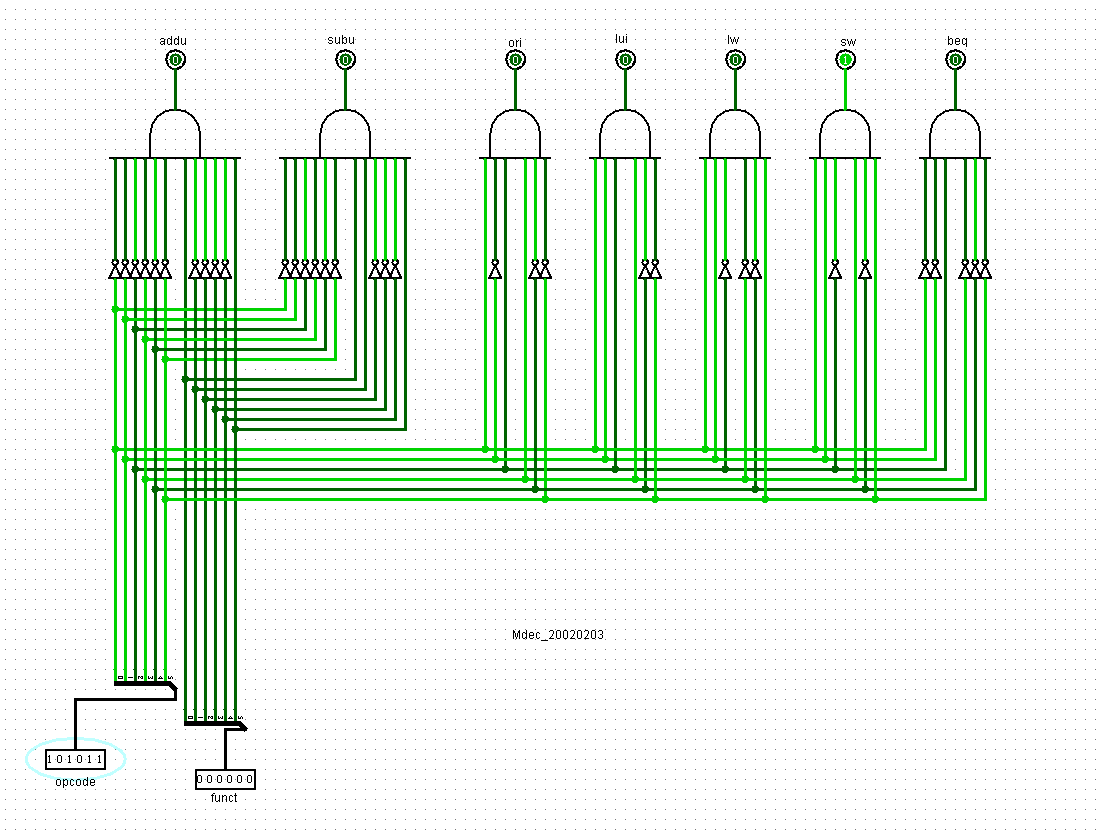
【lui】



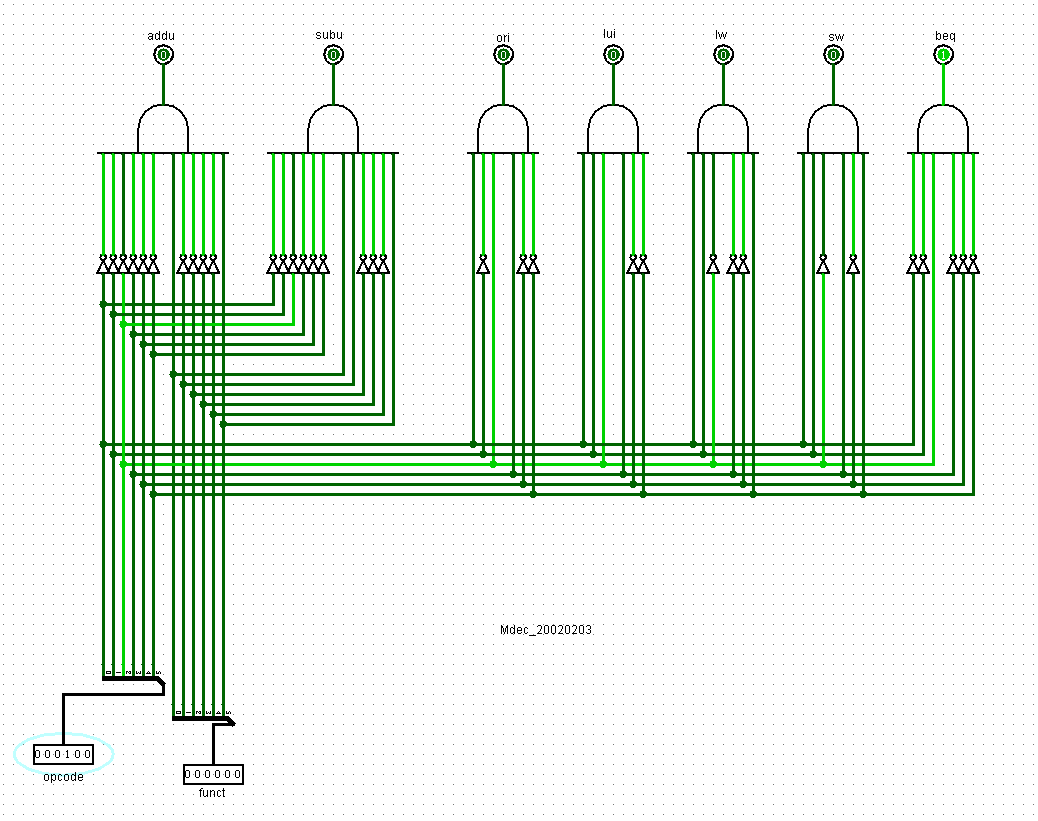
【lw】



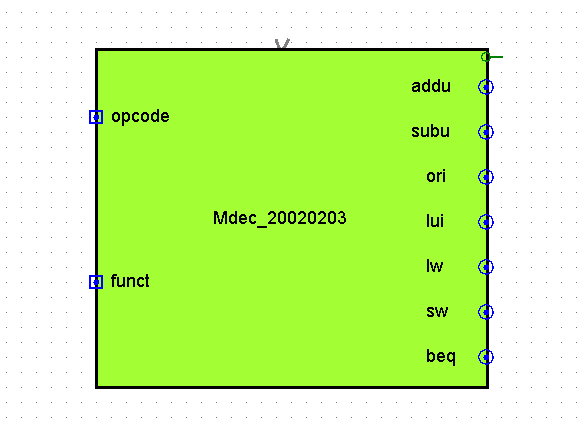
【sw】



【beq】



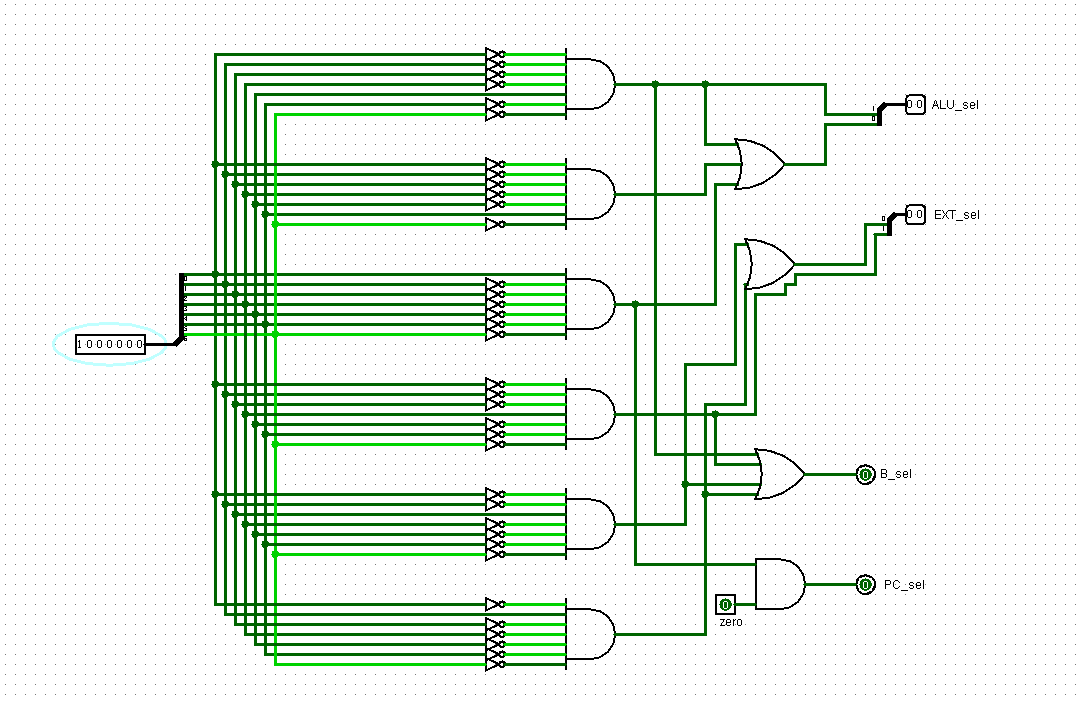
**3．编辑逻辑符号，备用。**



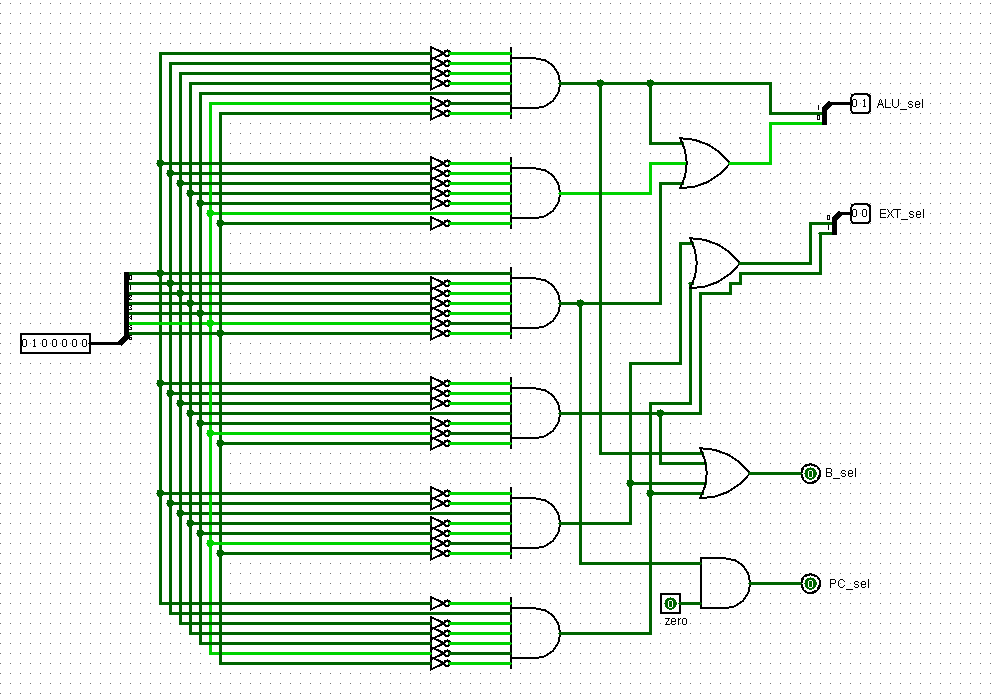
**五、相关选择信号的编码器设计与验证**

1、**在Logisim中设计实现编码器，进行验证，并截取正确结果电路图。**

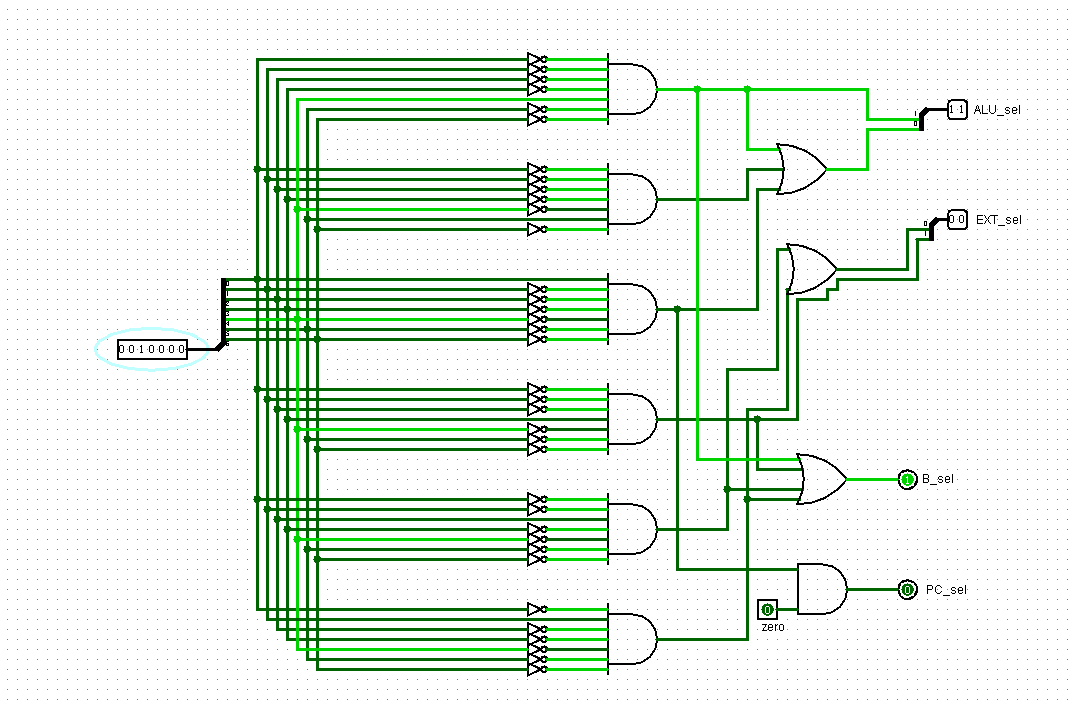
**【addu】**

****

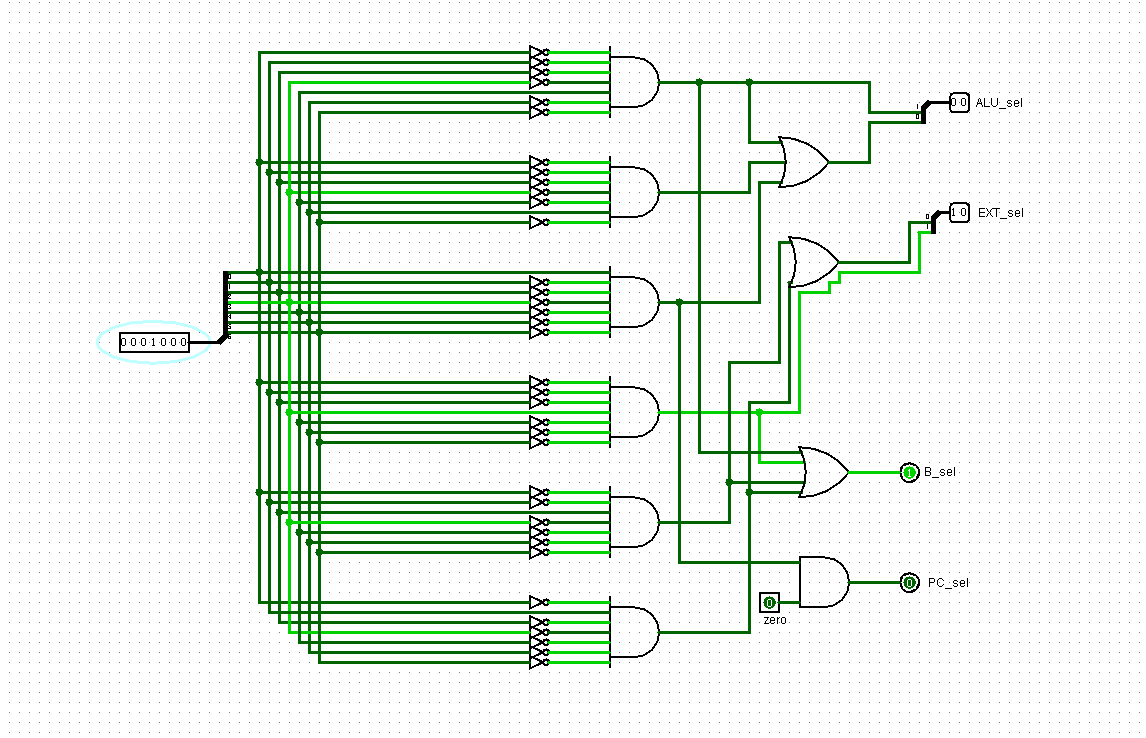
**【subu】**



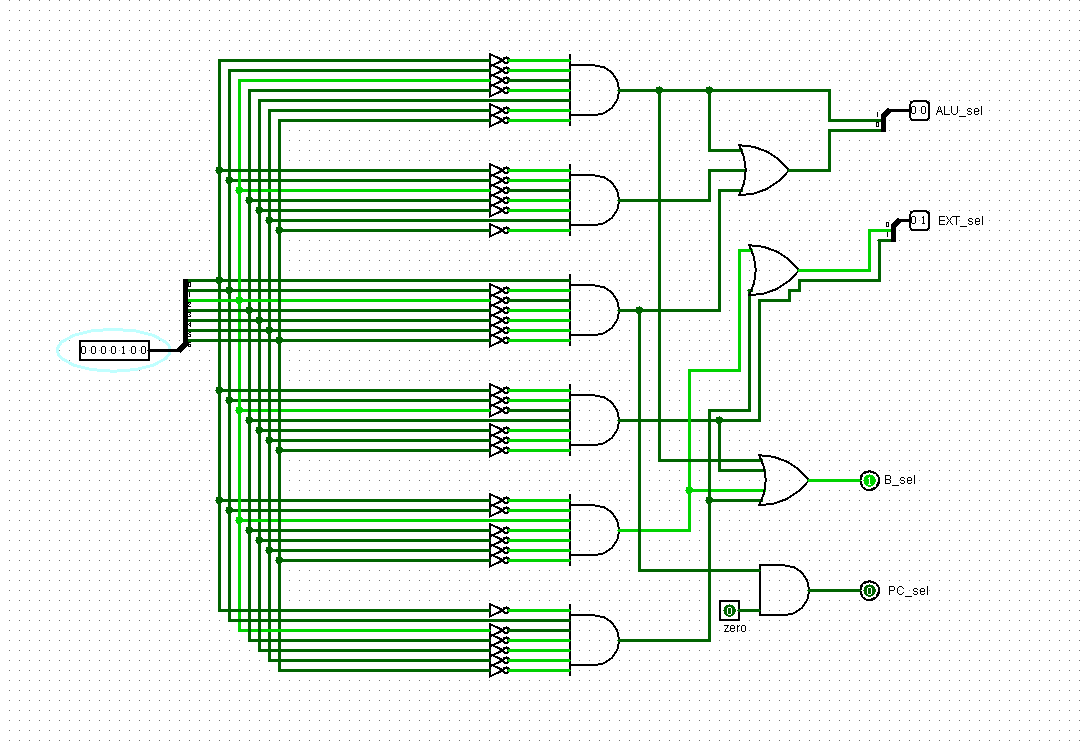
**【ori】**



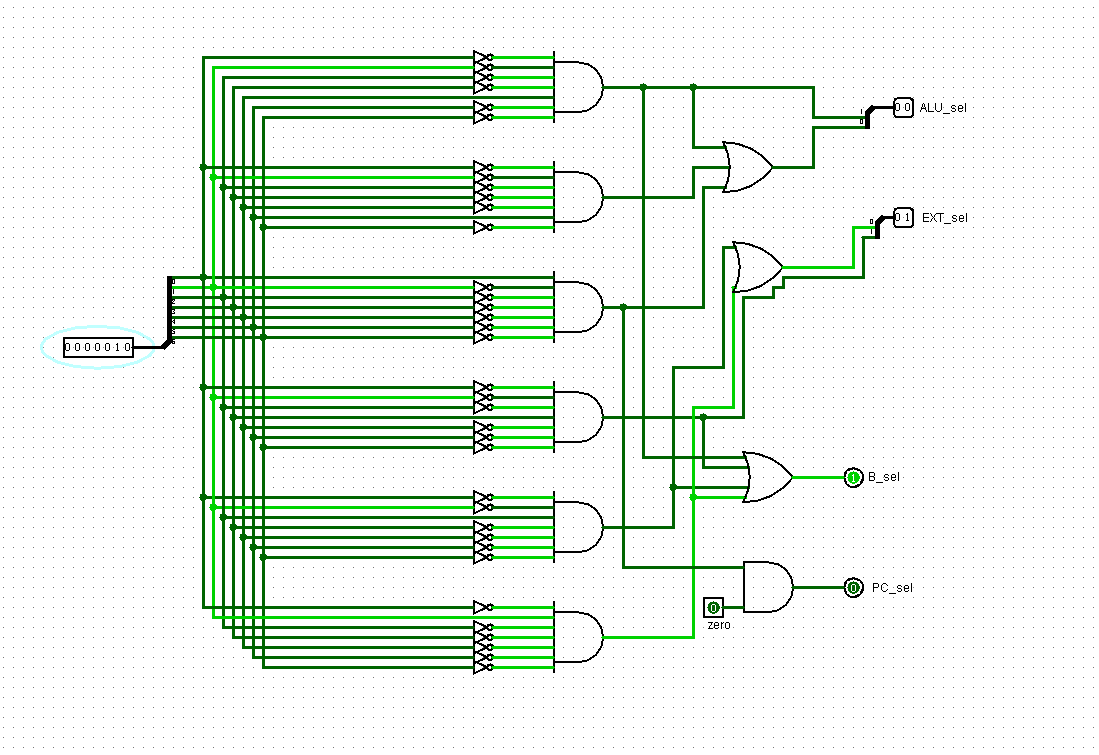
**【lui】**



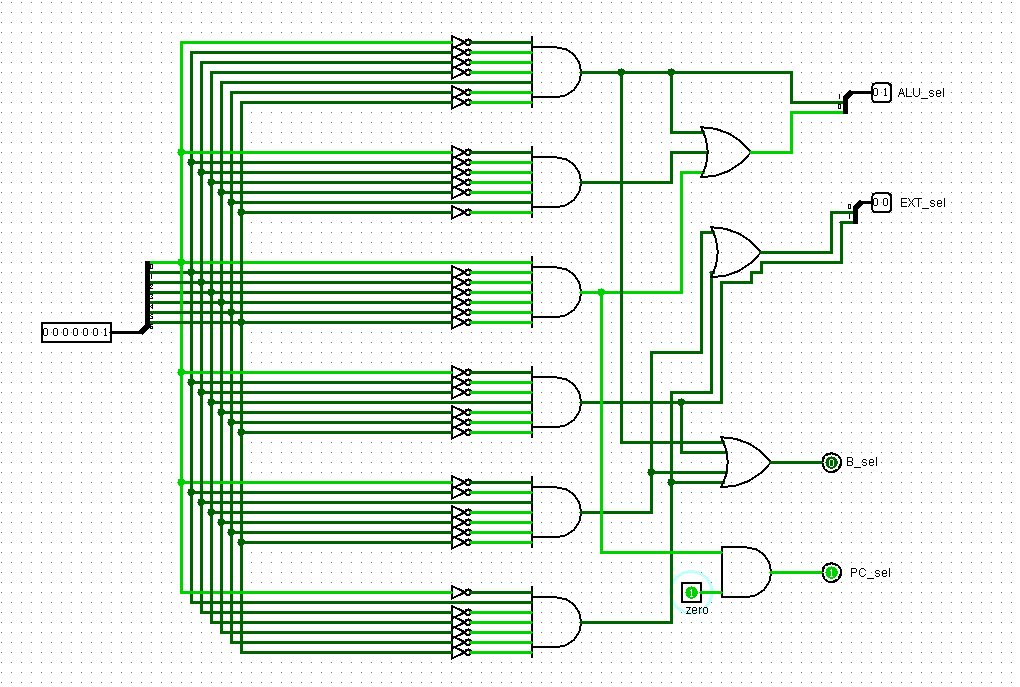
**【lw】**



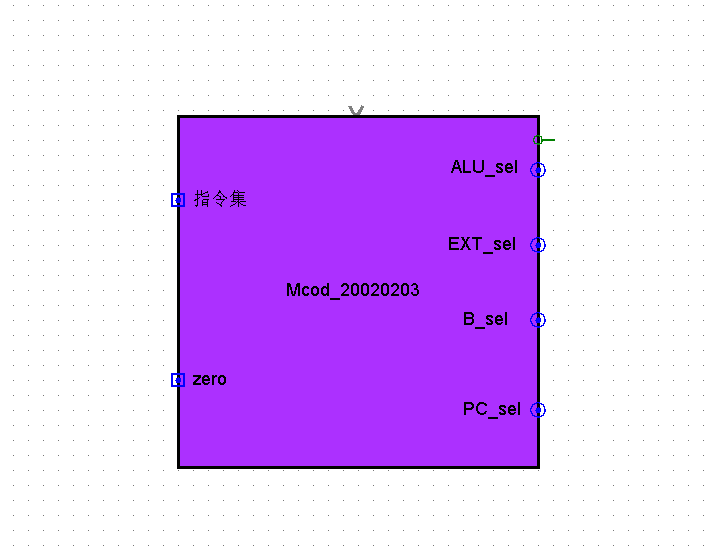
**【sw】**



**【beq】**

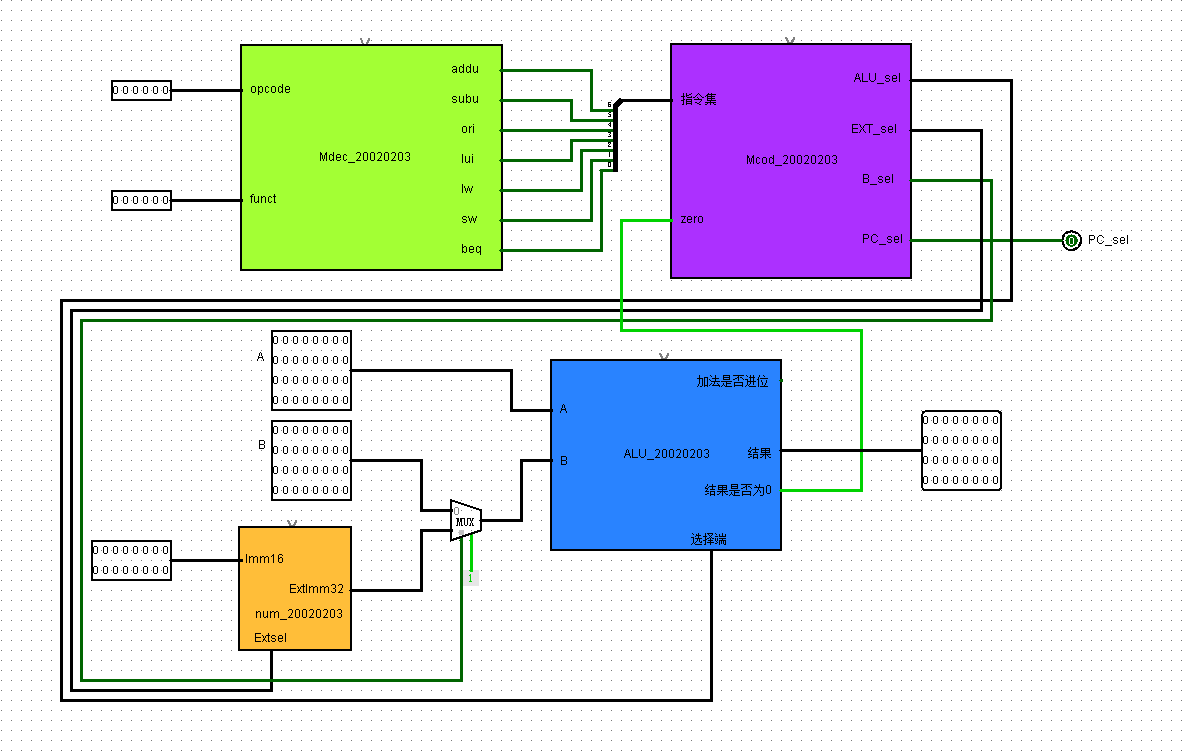


2、**编辑逻辑符号（Mcod-学号），备用。**



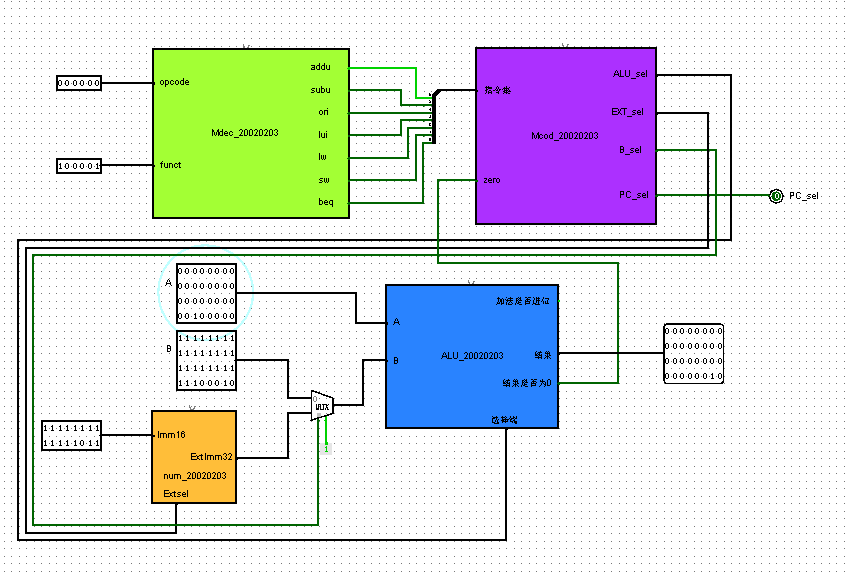
**六、综合验证**

**1．在Logisim中，按照图1进行电路（逻辑符号）连接。**

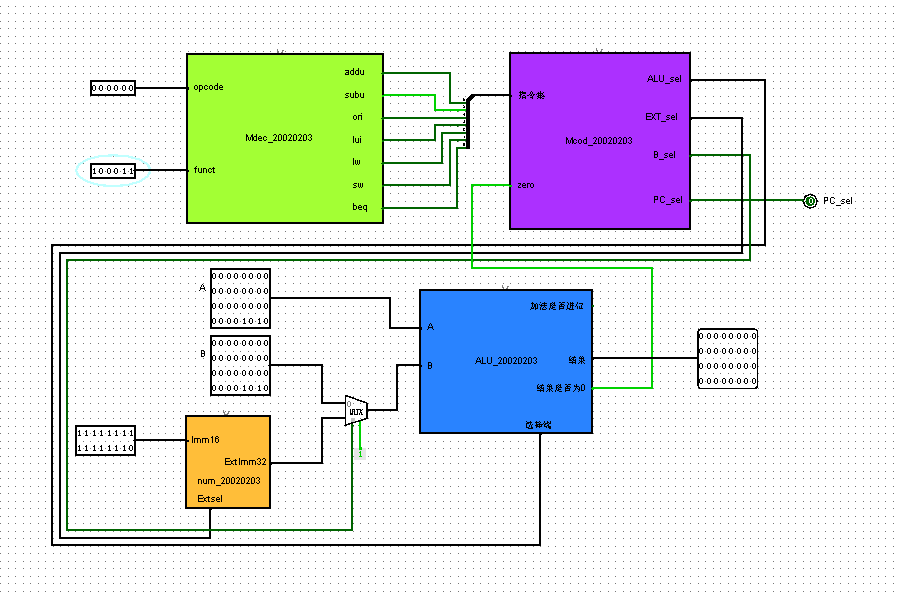


**2．按给定数据（注：十进制）进行测试并截取正确结果电路图。**

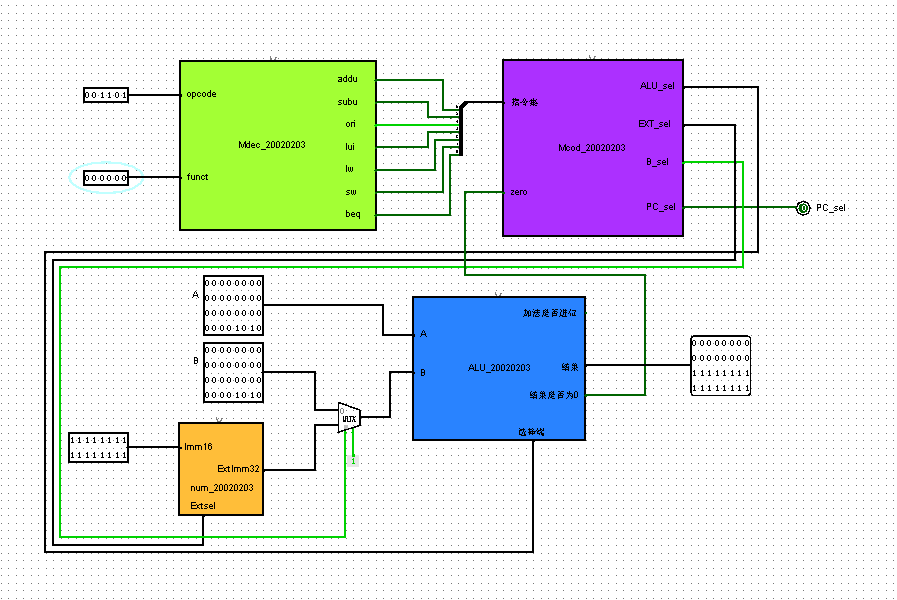
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | 二进制A | B | 二进制B | Imm16 | 二进制Imma16 | opcode | funct | 功能 | 结果 |
| 32 | 00100000 | -30 | 11100010 | -6 | 11111010 | 000000 | 100001 | addu | C:\Users\user\AppData\Local\Temp\1638083304(1).png |
| 10 | 00001010 | 10 | 00001010 | -2 | 11111110 | 000000 | 100011 | subu | C:\Users\user\AppData\Local\Temp\1638082368(1).png |
| 10 | 00001010 | 10 | 00001010 | -1 | 11111111 | 001101 | x | ori | C:\Users\user\AppData\Local\Temp\1638082448(1).png |
| 0 | 00000000 | 10 | 00001010 | 15 | 00001111 | 001111 | x | lui | C:\Users\user\AppData\Local\Temp\1638082502(1).png |
| 33 | 00100001 | 33 | 00100001 | 32 | 00100000 | 000100 | x | beq | C:\Users\user\AppData\Local\Temp\1638082668(1).png |
| 2 | 00000010 | 10 | 00001010 | 32 | 00100000 | 000100 | x | beq | C:\Users\user\AppData\Local\Temp\1638082878(1).png |
| 10 | 00001010 | 8 | 00001000 | -5 | 11111011 | 100011 | x | lw | C:\Users\user\AppData\Local\Temp\1638083036(1).png |

1. **当A=32、B= -30、Imm16=-6时，完成addu测试，截取测试结果并说明；**

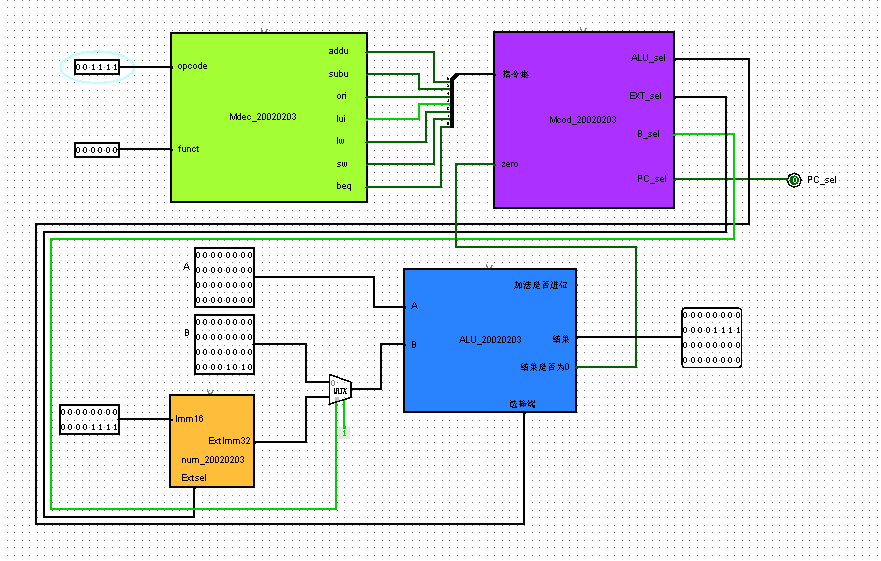
**(2)当A=10、B=10、Imm16=-2时，完成subu测试，截取测试结果并说明；**



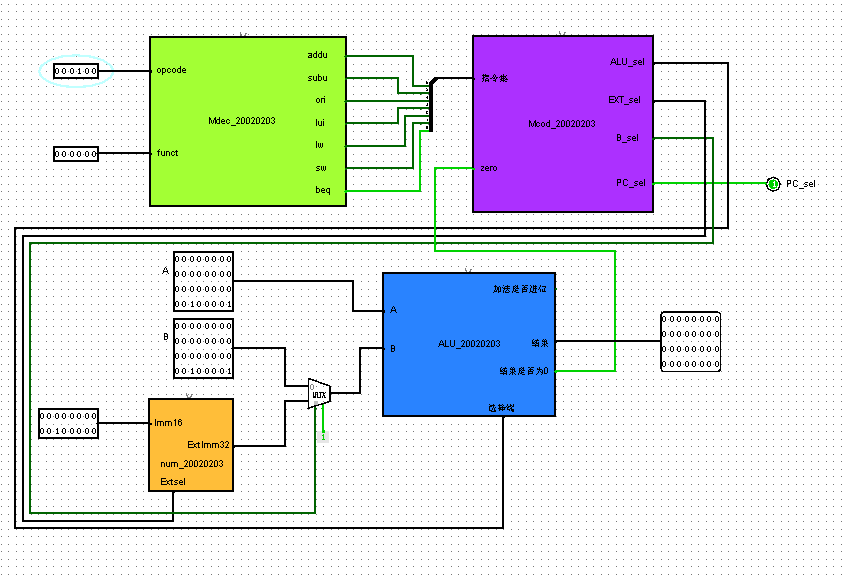
**(3)当A=10、B=10，Imm16=-1时，完成ori测试，截取测试结果并说明；**



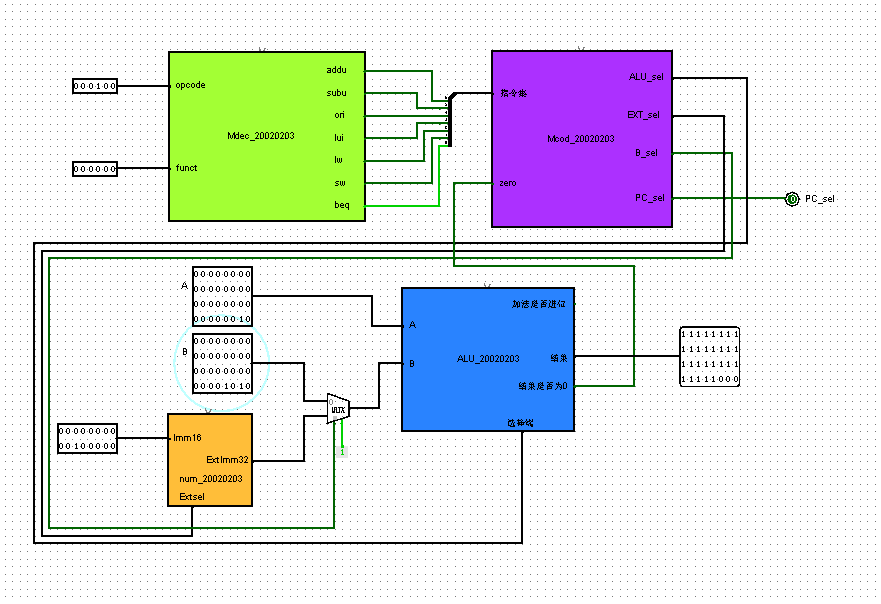
**(4)当A=0、B=10，Imm16=15时，完成lui测试，截取测试结果并说明；**



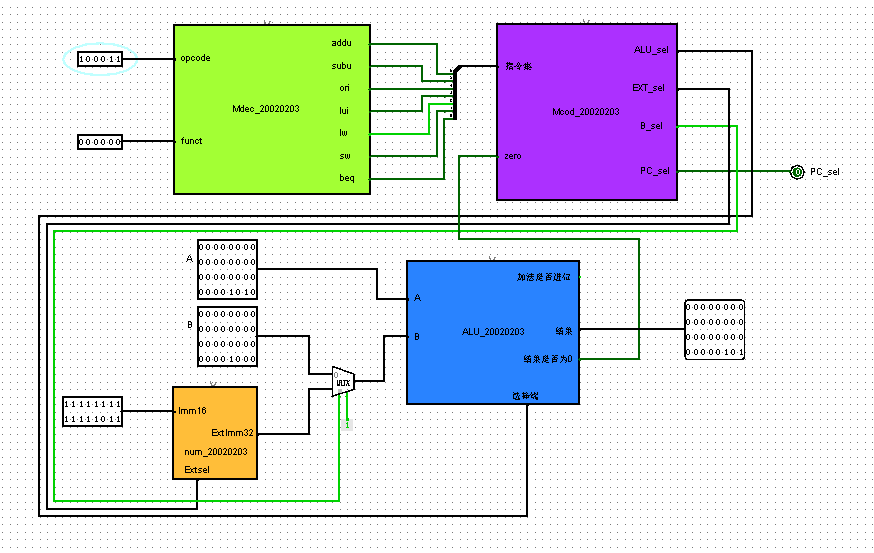
**(5)当A=33、B=33，Imm16=32时，完成beq测试，截取测试结果并说明；**



**(6)当A=2、B=10，Imm16=32时，完成beq测试，截取测试结果并说明；**



**(7)当A=10、B=8，Imm16=-5时，完成lw测试，截取测试结果并说明；**



**七、综合以上内容，完成WORD报告汇集并进行小结。**

本次大作业我按照要求完成了ALU、MIPS、立即数扩展器、编码器等器件。32位加法器、减法器是logisim的基本器件，但我们这次通过使用一位加法器不断串联，得到了自制的32位加法器，减法器可以通过加法器实现，原理是A-B=A+[B]补码，故完成了加法器与减法器的自行设计。在设计过程中，当我使用verilog硬件描述语言实现ALU的时候我注意到，&和&&是不一样的，&是按位与，而&&是逻辑与，这是本次设计过程中记忆较为深刻的点。在设计后面几个器件的时候，我还学会了几个新的器件的使用，分别为：分线器、数据选择器、位数扩展器等，我认为最有用的是‘constant’常量这个器件，它可以给元件一个常量输入，这样就可以使某个输入始终保持一个固定的值，不受外界输入干预。最后按照要求将几个器件连起来时，让我的逻辑一下变得清晰起来，即这次设计主要就是分为运算模块ALU，发送指令模块，译码模块和立即数扩展模块。当指令模块发出指令后，指令被译为对应的选择信号，这些选择信号控制着输入和ALU的运算两个部分，最后经ALU运算输出对应结果。

通过这次实验，我不仅更加熟练的掌握了logisim软件的使用，而且明白了简易 “CPU”的结构、工作原理，这让我思维更加清晰，对数字逻辑这门课也有了更深的认识，真正做到了将学习的知识用到实际生活当中去。