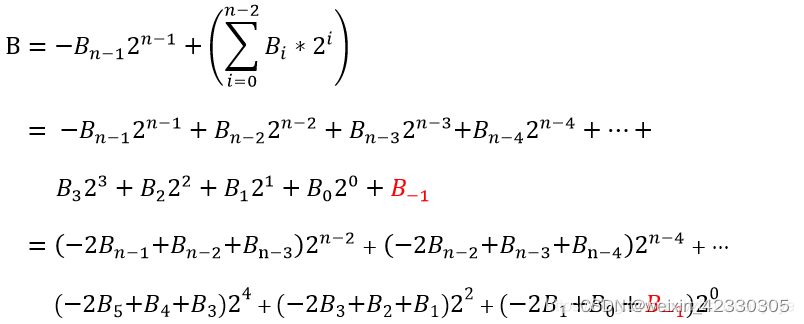
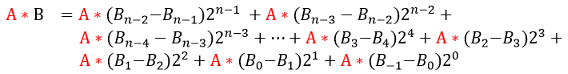
快速乘法器设计

摘要：报告阐述了快速乘法器设计。通过采用***radix-4 Booth***编码以及***Wallace*加法树**方法，实现设计32位无符号整型**快速乘法器**。并通过代码改进，实现**32位有符号乘法**功能，使用者可根据实际情况选用。并在此基础上，设计出**4级流水**的Booth编码器改进，**2级流水**的**符合*IEEE*标准**的**32位单精度浮点型乘法器**，和**改进的*radix-16 Booth***编码改进三种创新。

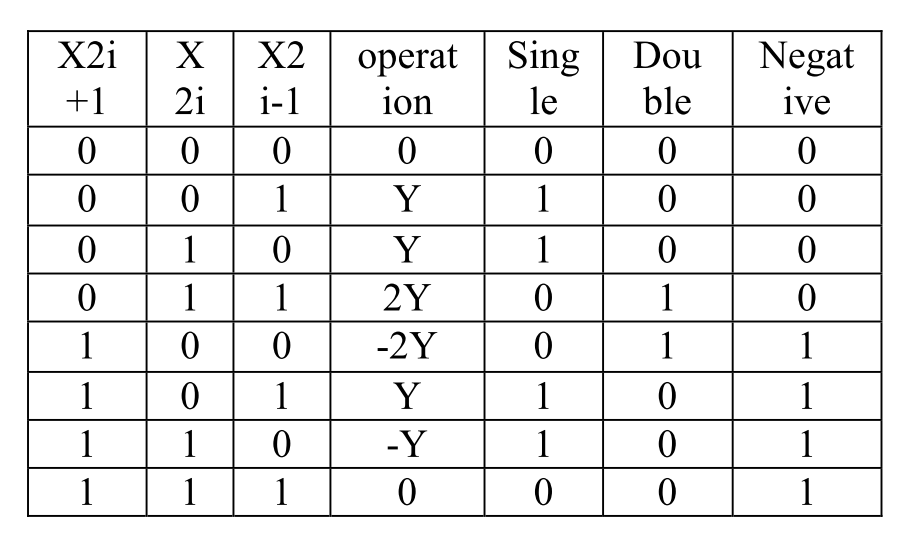
关键词：**radix-4 Booth**；**Wallace加法树**；**有符号**；**4级流水；IEEE标准**；**单精度浮点型乘法器**；**改进的radix-16 Booth**

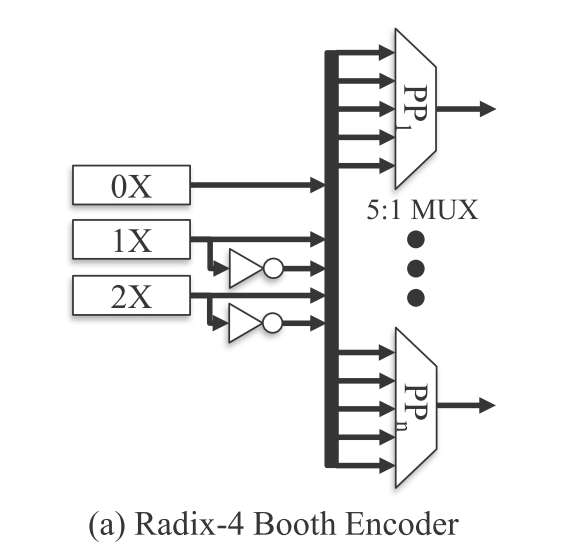
# 一、基本功能实现-32位有无符号整型快速乘法器

采用radix4 Booth编码 ,根据计算公式

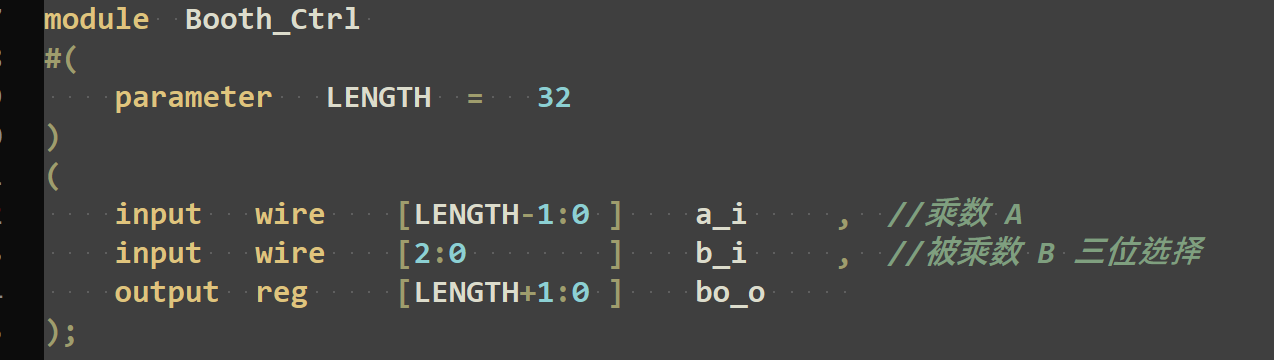
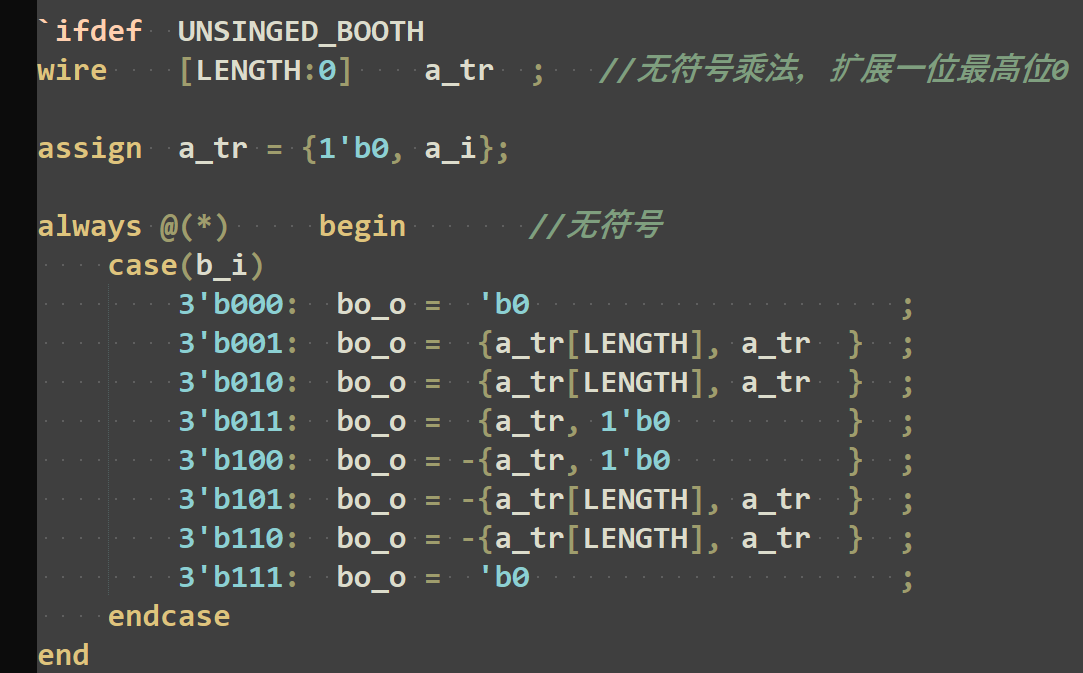
可以得到：

所以可以得到一下查找表：



根据此查找表[1]，构筑出部分积的多路选择器[2]：

创建一个转Booth编码的控制模块

1. a\_i为乘数A输入
2. b\_i 为乘数B的三位输入选择
3. bo\_o为booth编码后的值

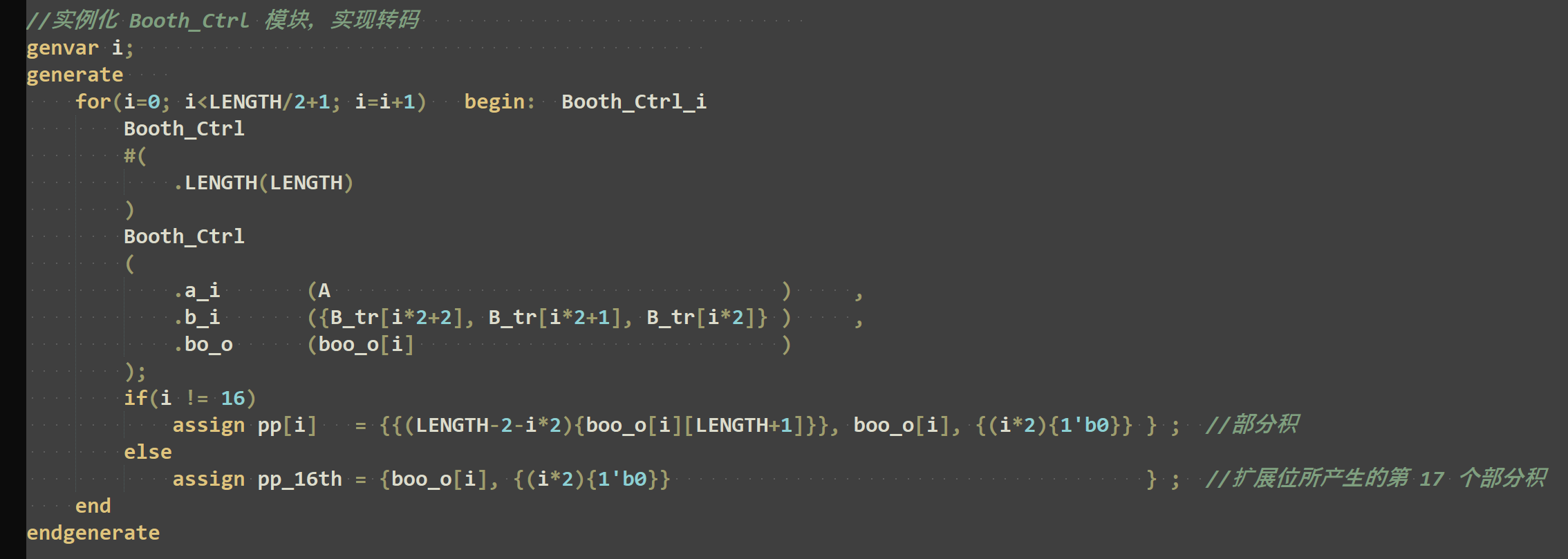
**Booth转码，对应上图查找表**

由于Booth编码基于补码有符号公式，所以转成无符号需要对其扩展最高位0。

所以轻易可得以下有符号编码：

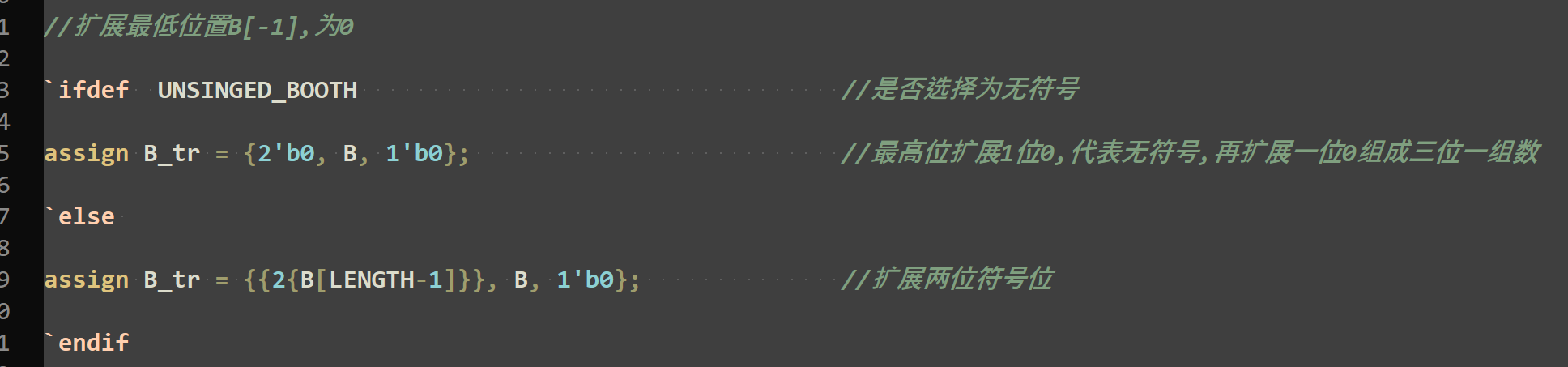
（注：此时bo\_o位宽可以减少两位，但为了方便与无符号对应，故而采取扩展两位符号位策略，结果不变）

转Booth编码的控制模块完成

顶层文件例化转Booth编码

生成17个部分积

第17个部分积由于B的低0位和两位符号位扩展而生成。



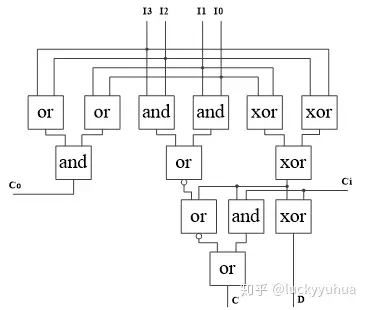
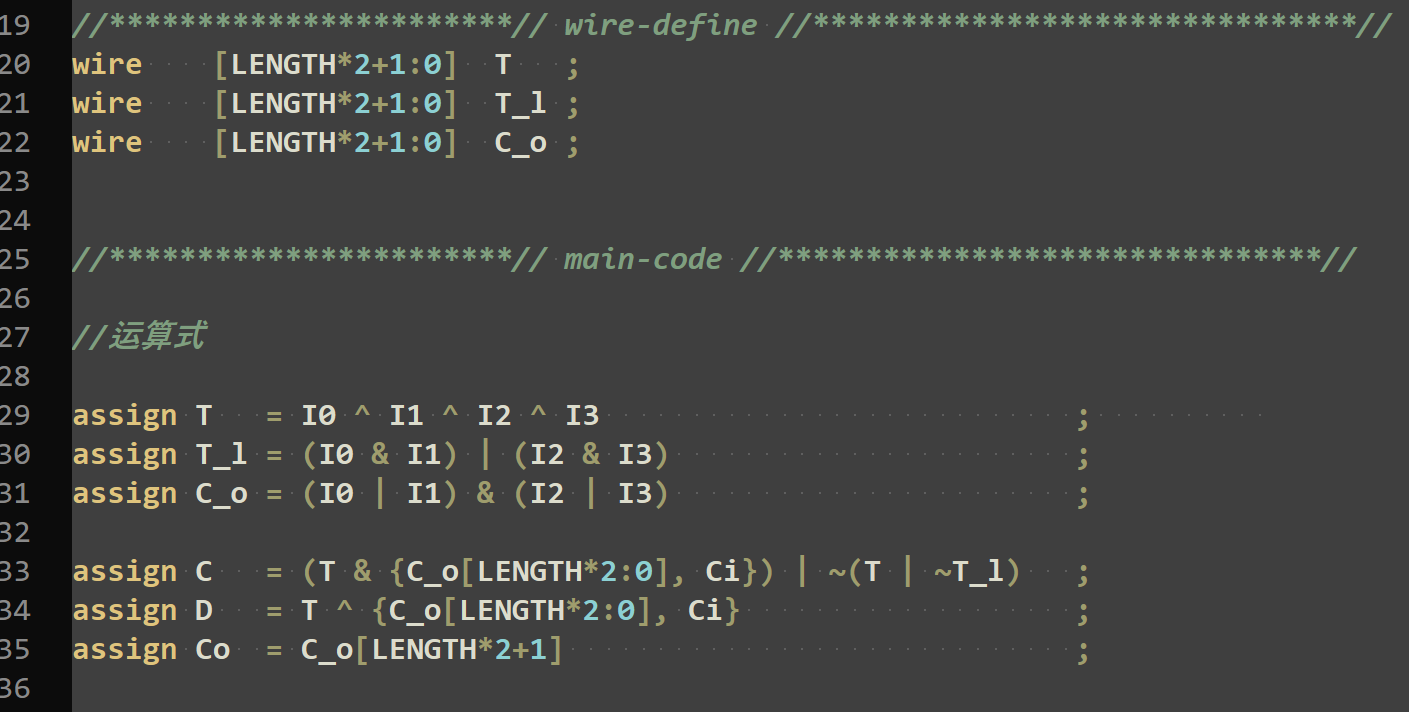
生成部分积后，相加则可得到结果，但由于行波进位加法器进位传递延时对电路性能存在非常大的影响，此时采用多级4-2压缩器组成Wallace加法树（也可以使用CSA），并行计算，实现优化运算。



4-2压缩器模块：



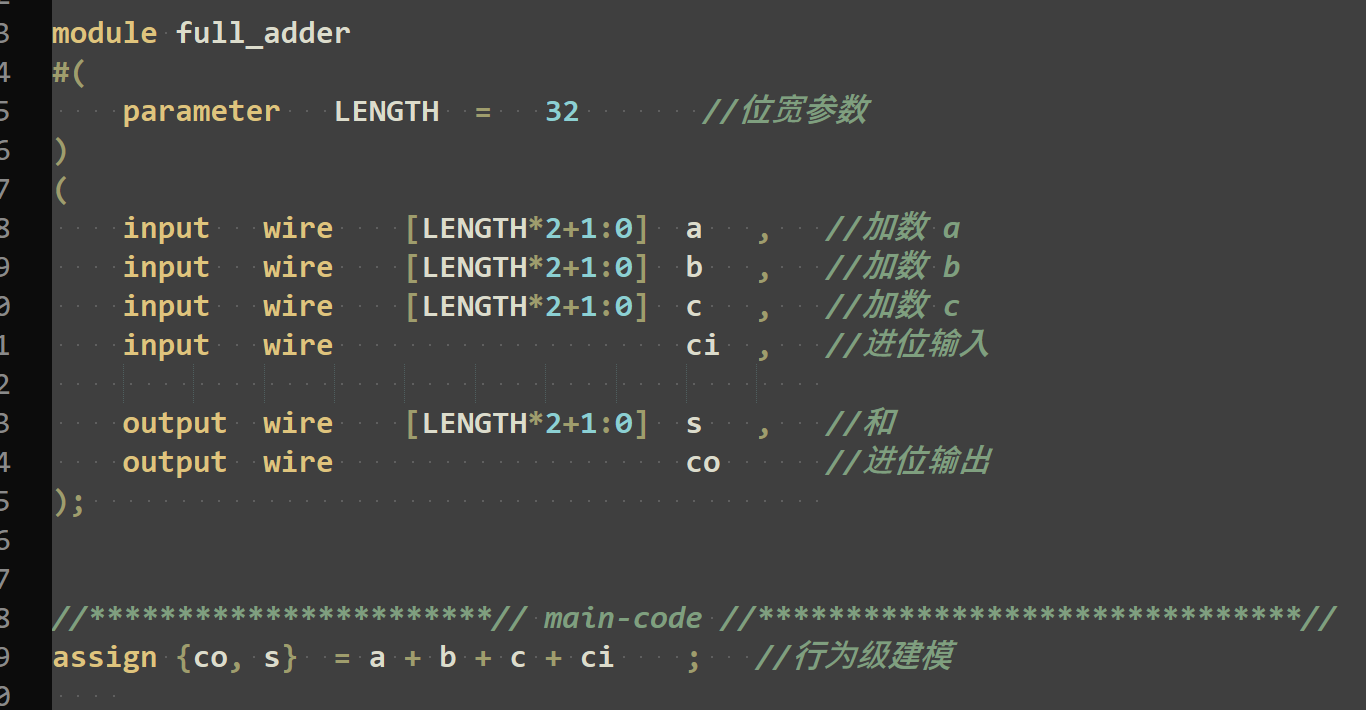
输入输出定义

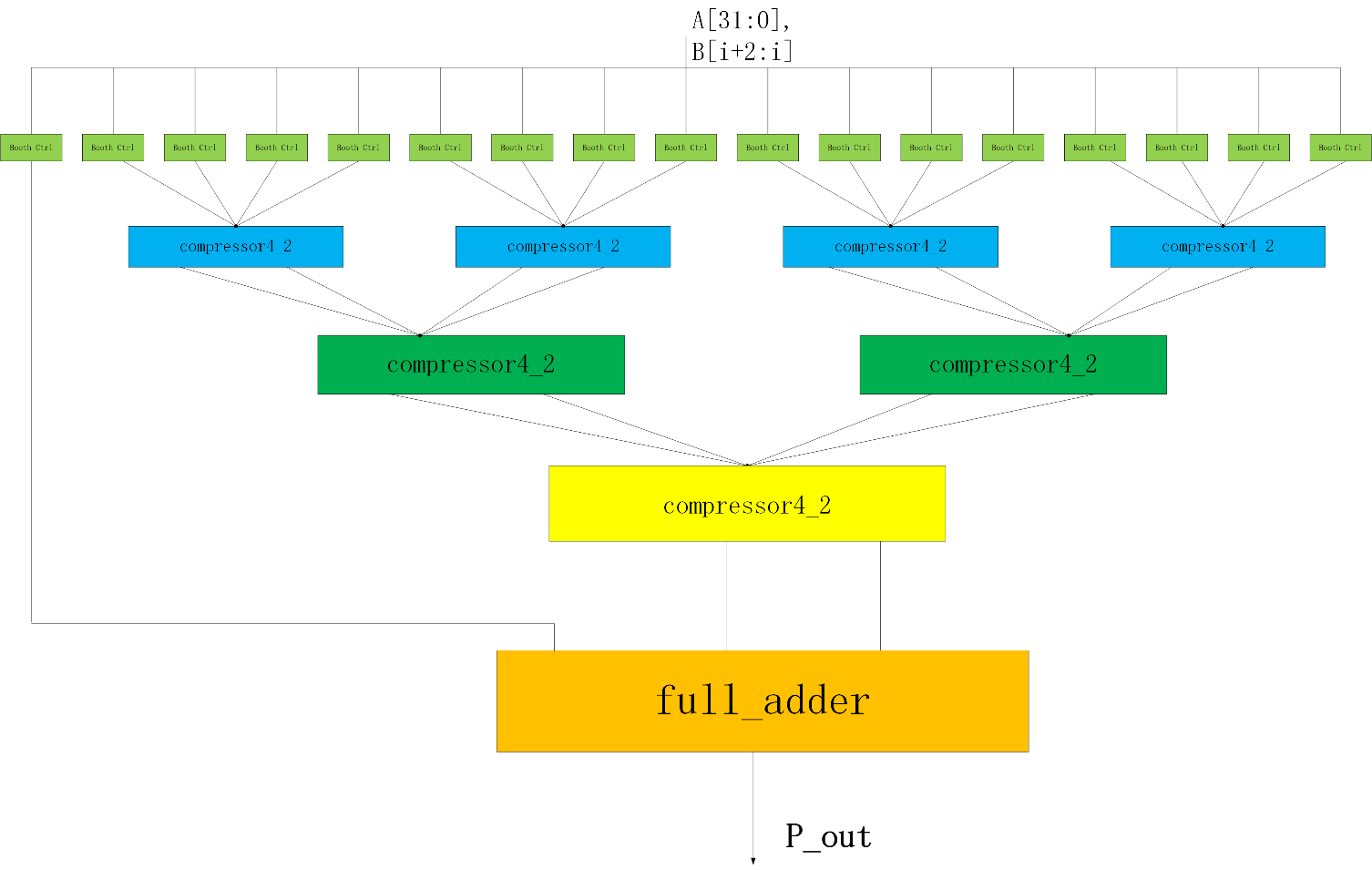
根据此逻辑图，得到以下代码：

4-2压缩器模块完成

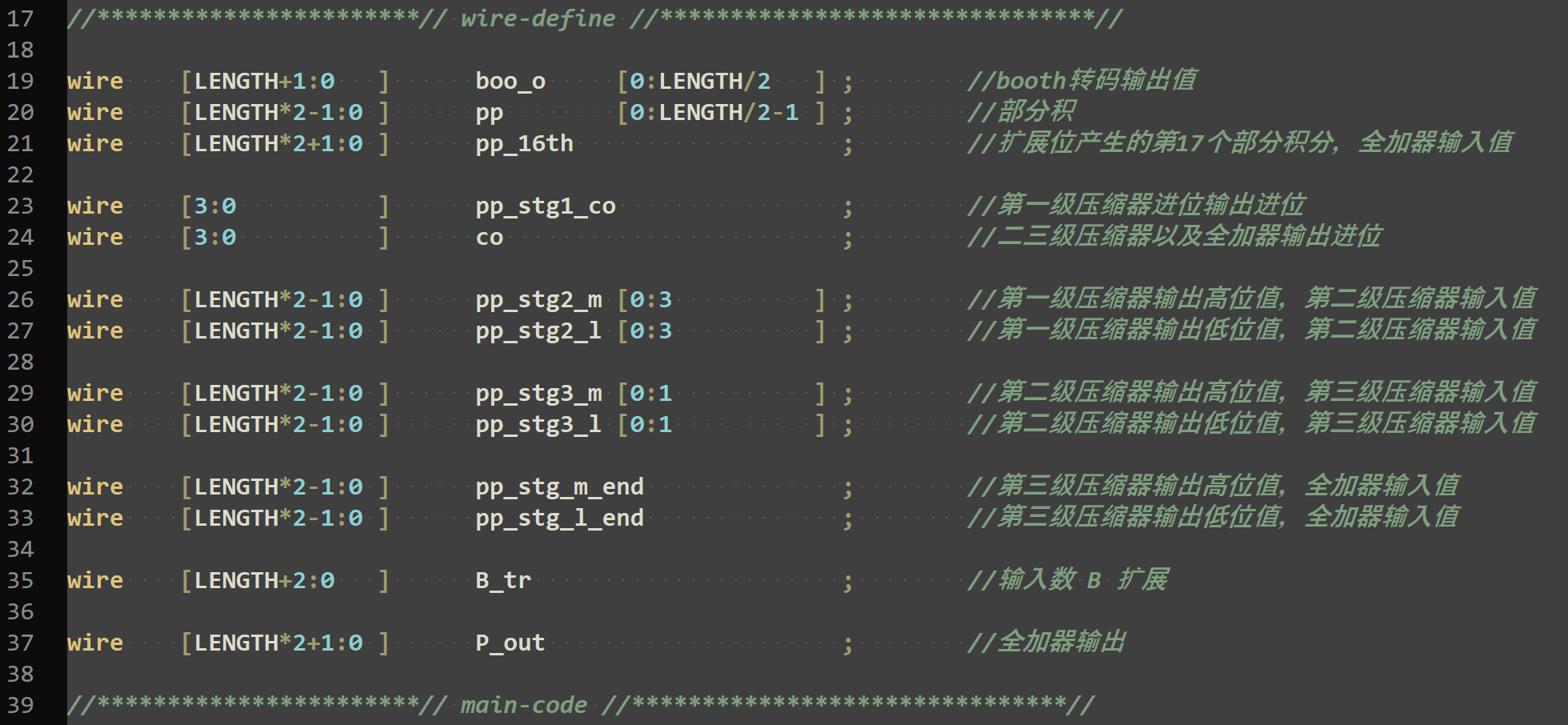
最后结果仍然需要一个全加器模块生成

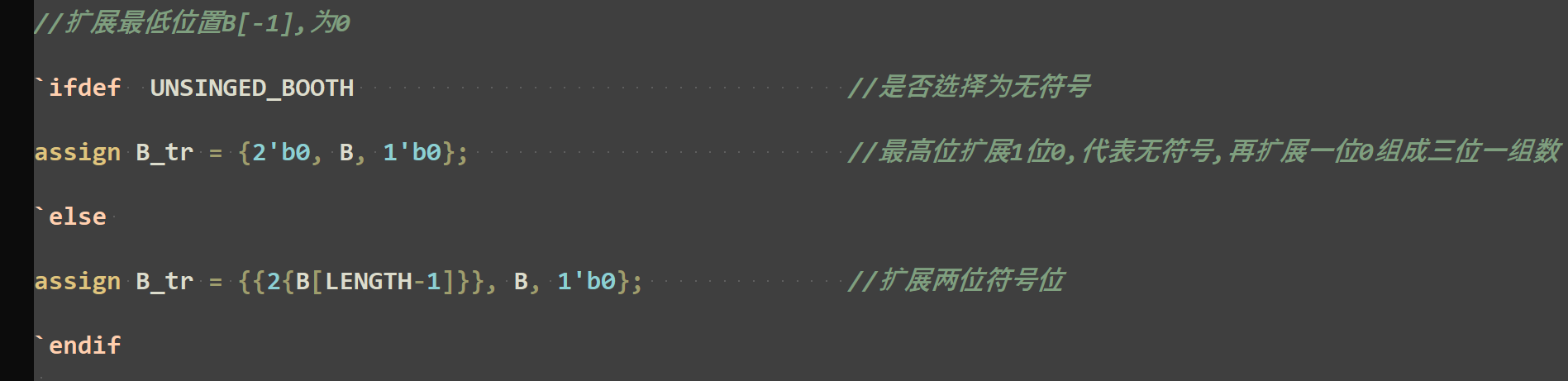
创建一个32位全加器：

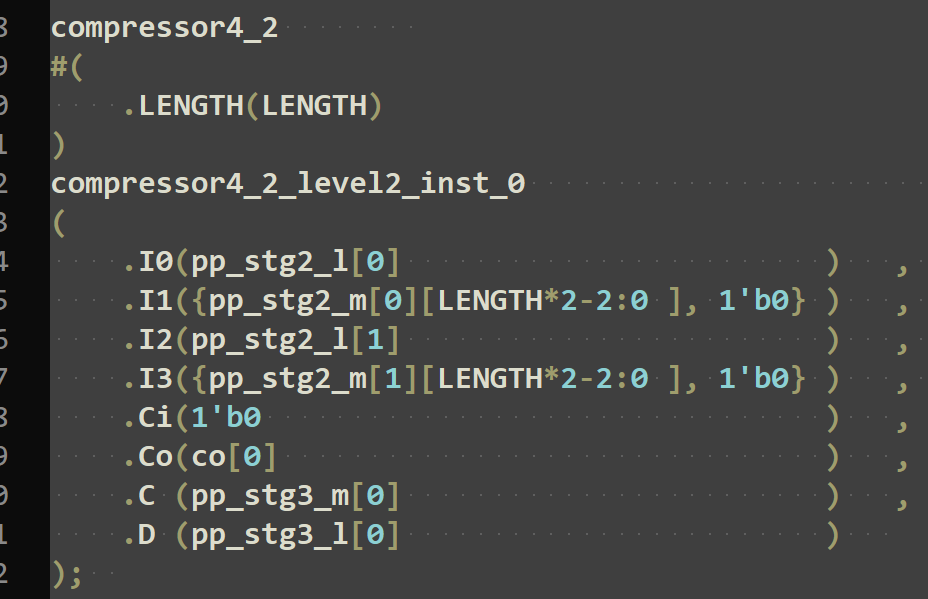
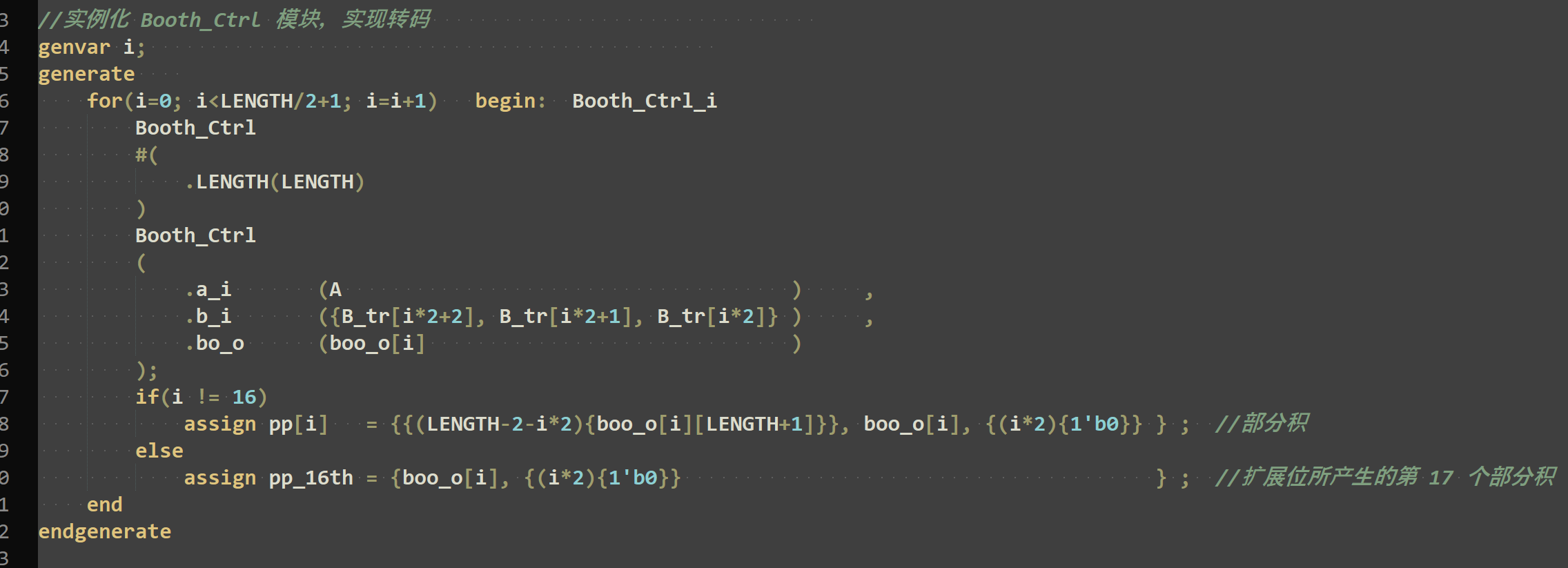
这里有3个输入，原因是前面4-2压缩器处理了16个部分积，还有因扩展生成的第17个部分积未处理，在这里加上。

所以得到以下流程图：

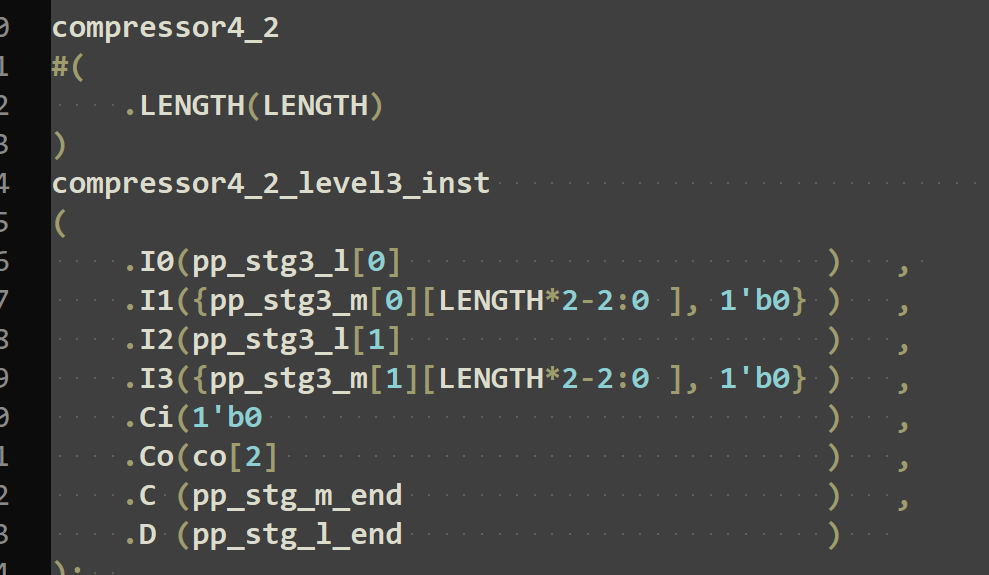
总共采用17个Booth\_Ctrl模块生成部分积，4个一级4-2压缩器，2个二级4-2压缩器，1个三级4-2压缩器，和一个全加器。

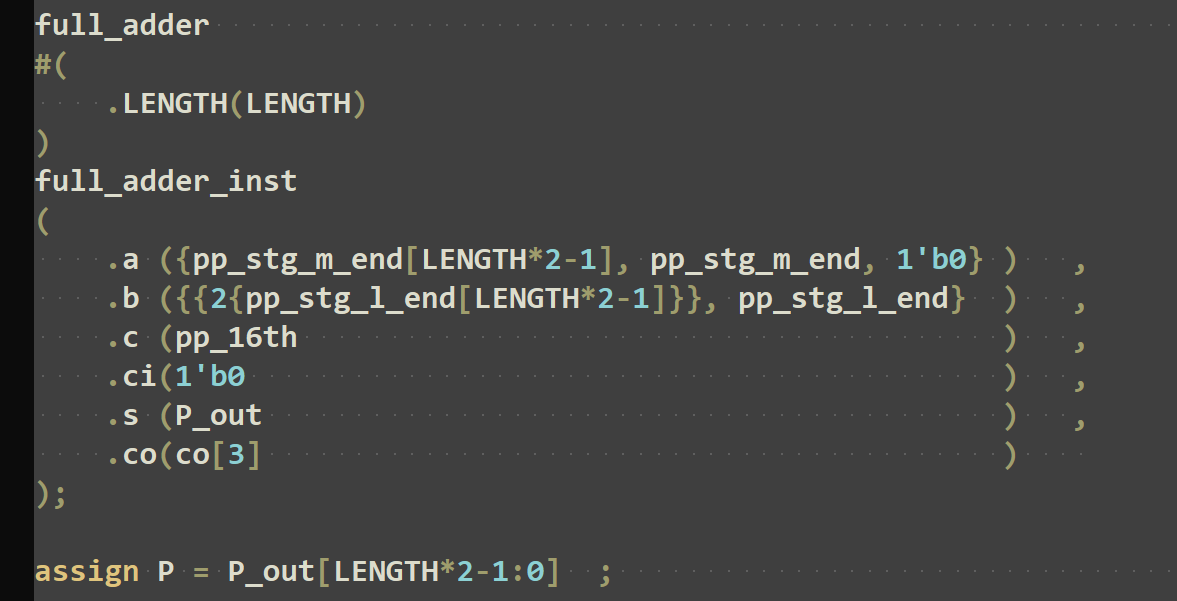
最终顶层文件：



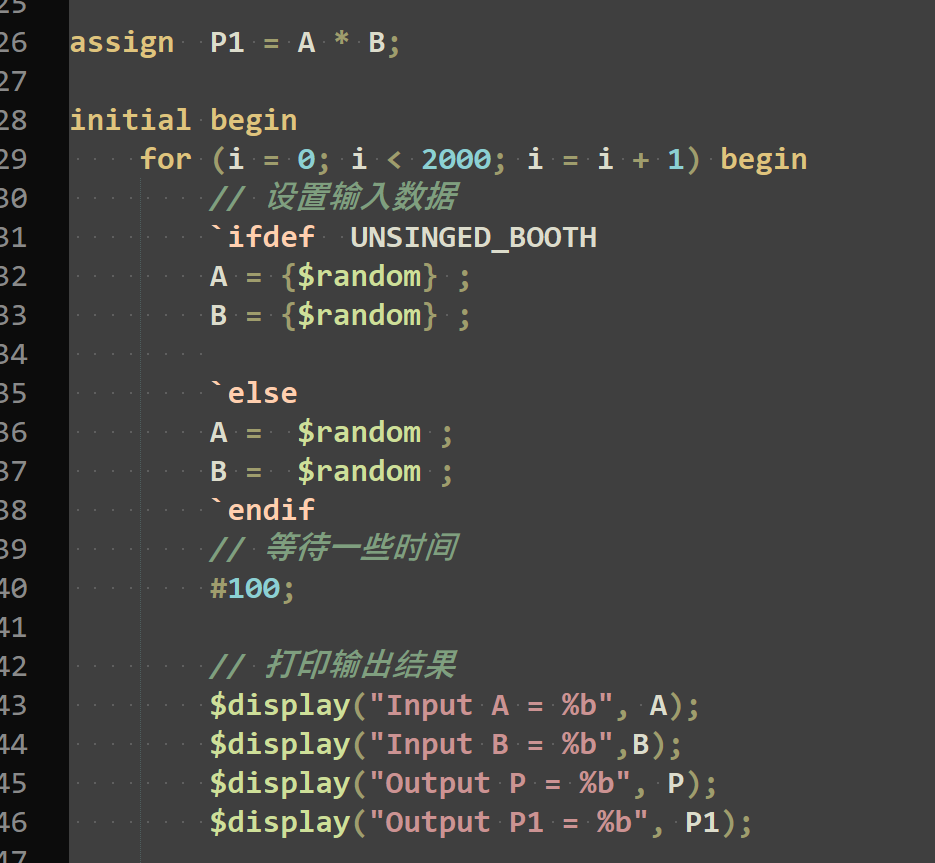
第一级，4个压缩器（上图）

第2级，2个压缩器（上图）



第3级，1个压缩器（上图）

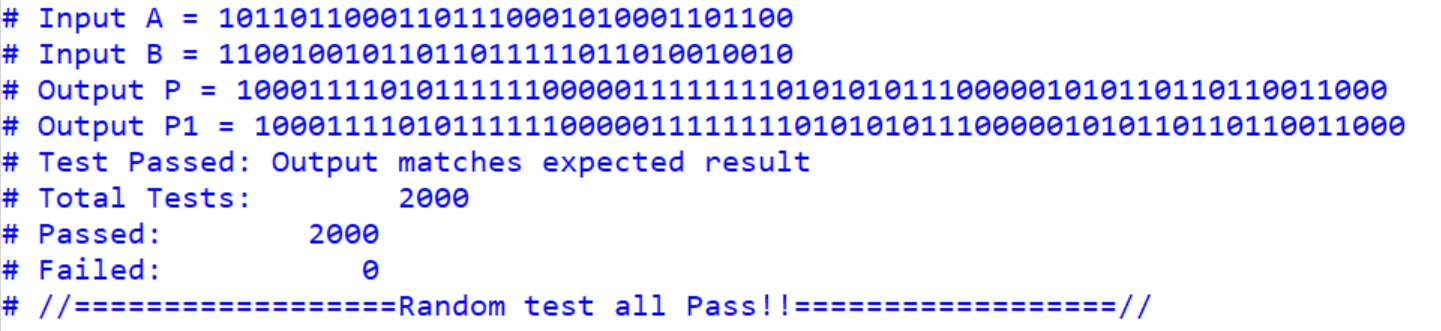
最后的全加器以及输出结果（上图）

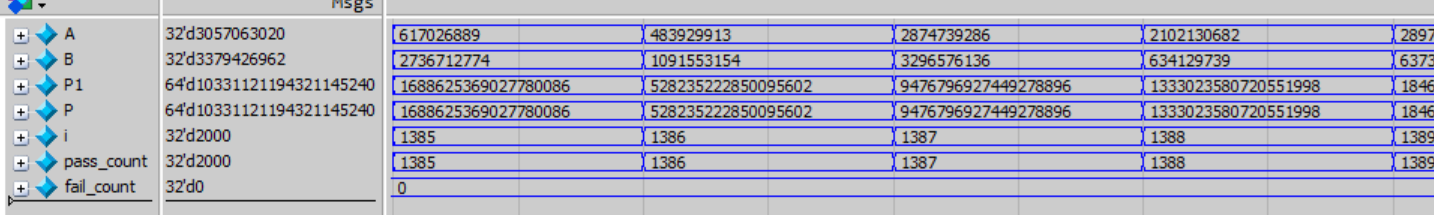
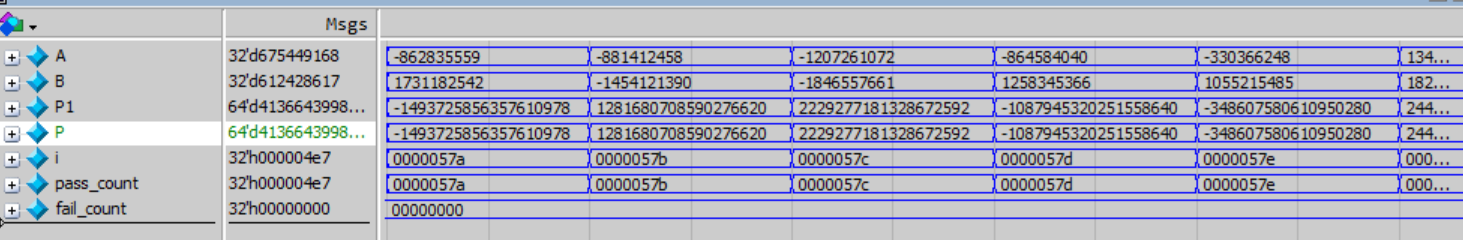
TB文件编写：

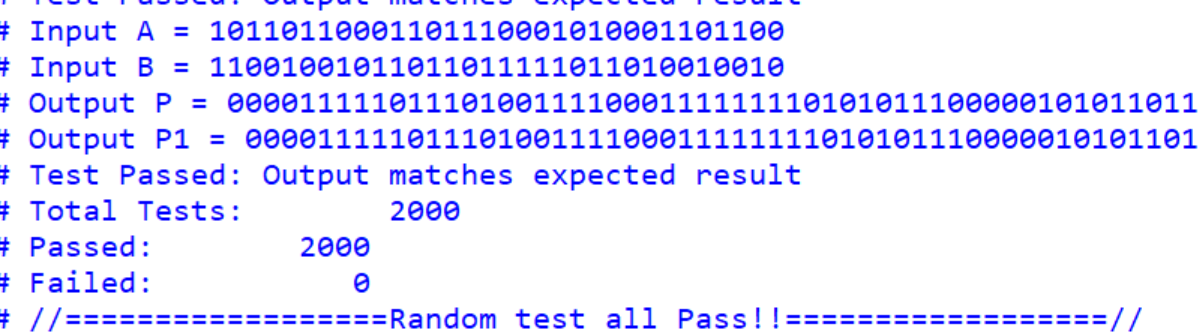
根据条件编译，选择有符号还是无符号，对应前面有无符号Booth编码模块。

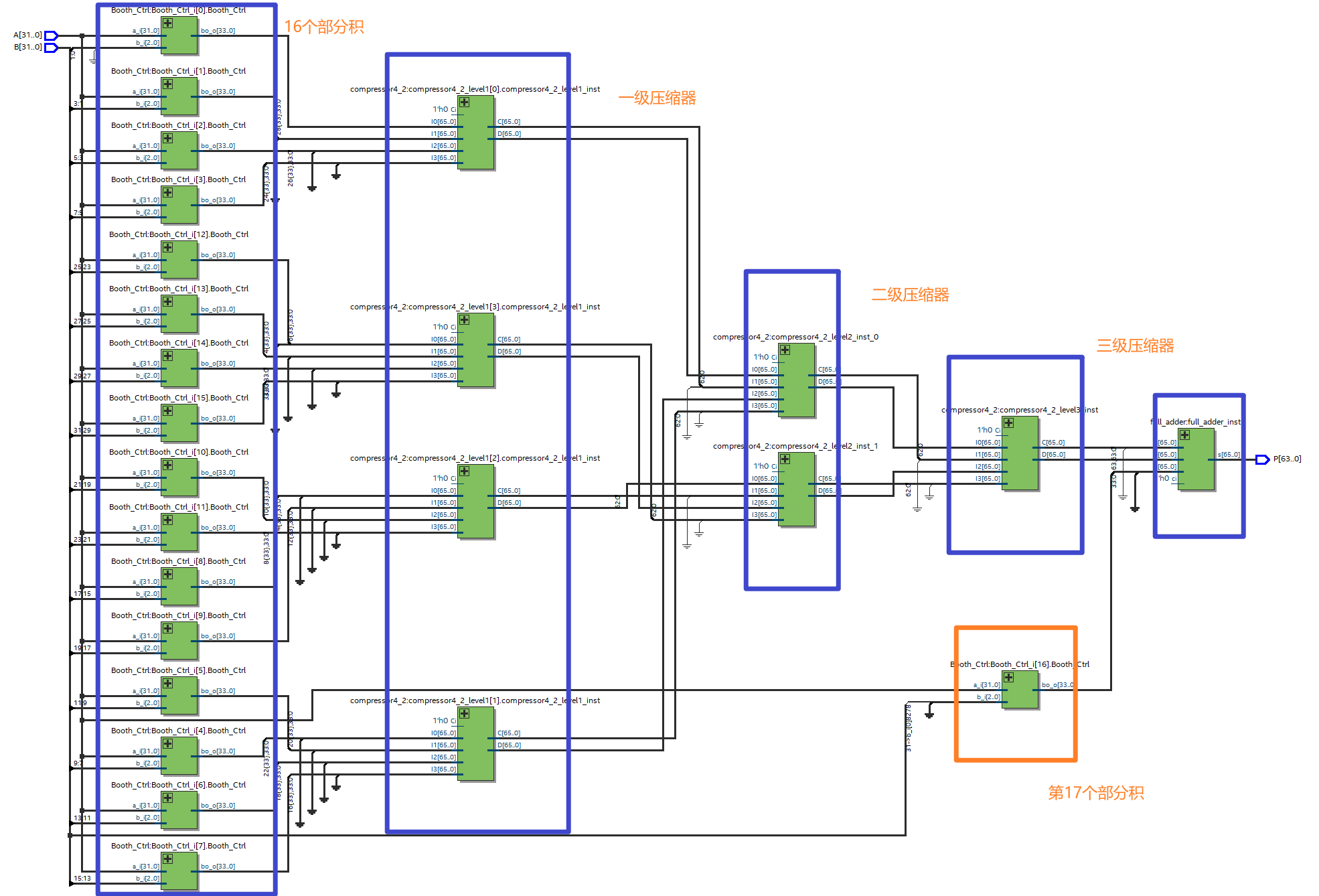
随机生成2000个数相乘，得到结果与P1对比（P1为调用软件自带乘法运算结果）

使用quartus ii 18.0创建工程，并联合modelsim仿真

仿真结果以及波形如下：

有符号乘法仿真结果以及波形：

成功验证乘法器。

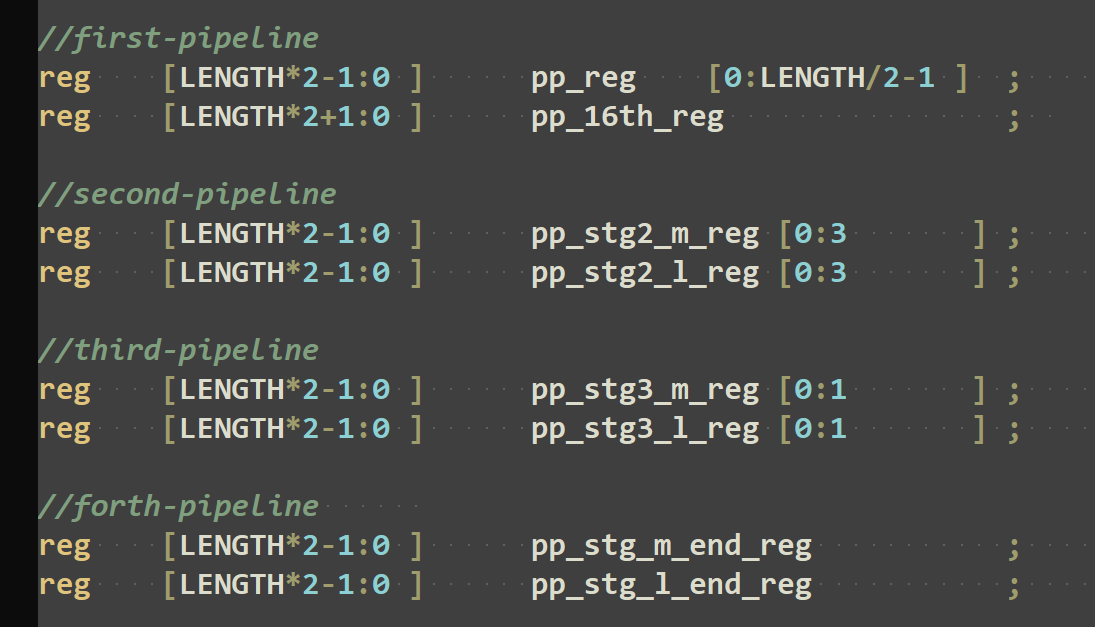
查看RTL视图：

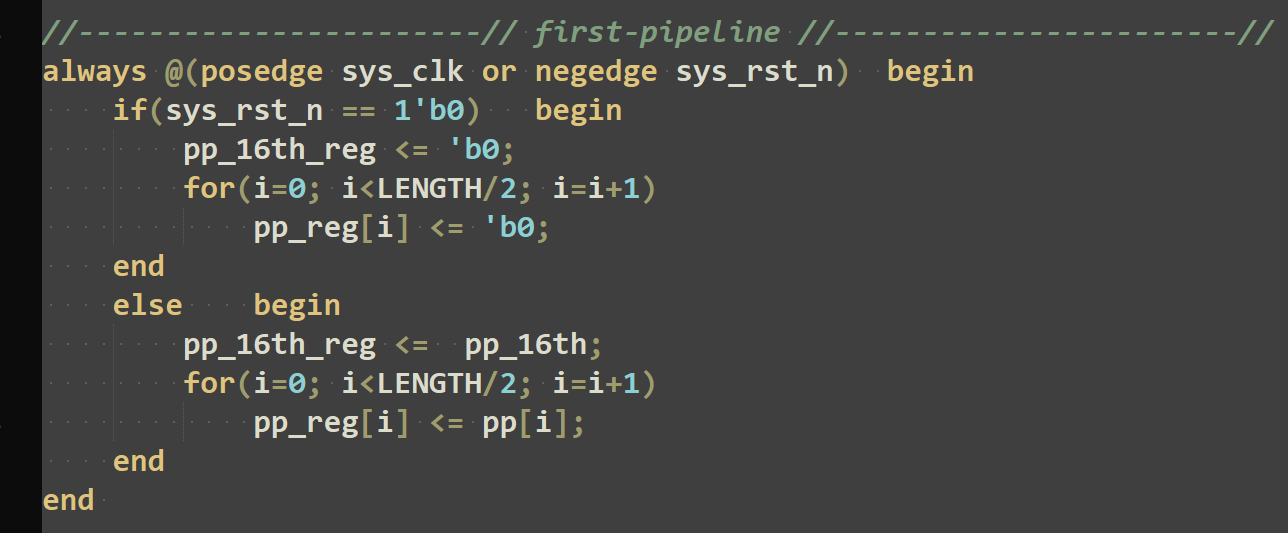
32位**有无**符号快速乘法器设计成功。

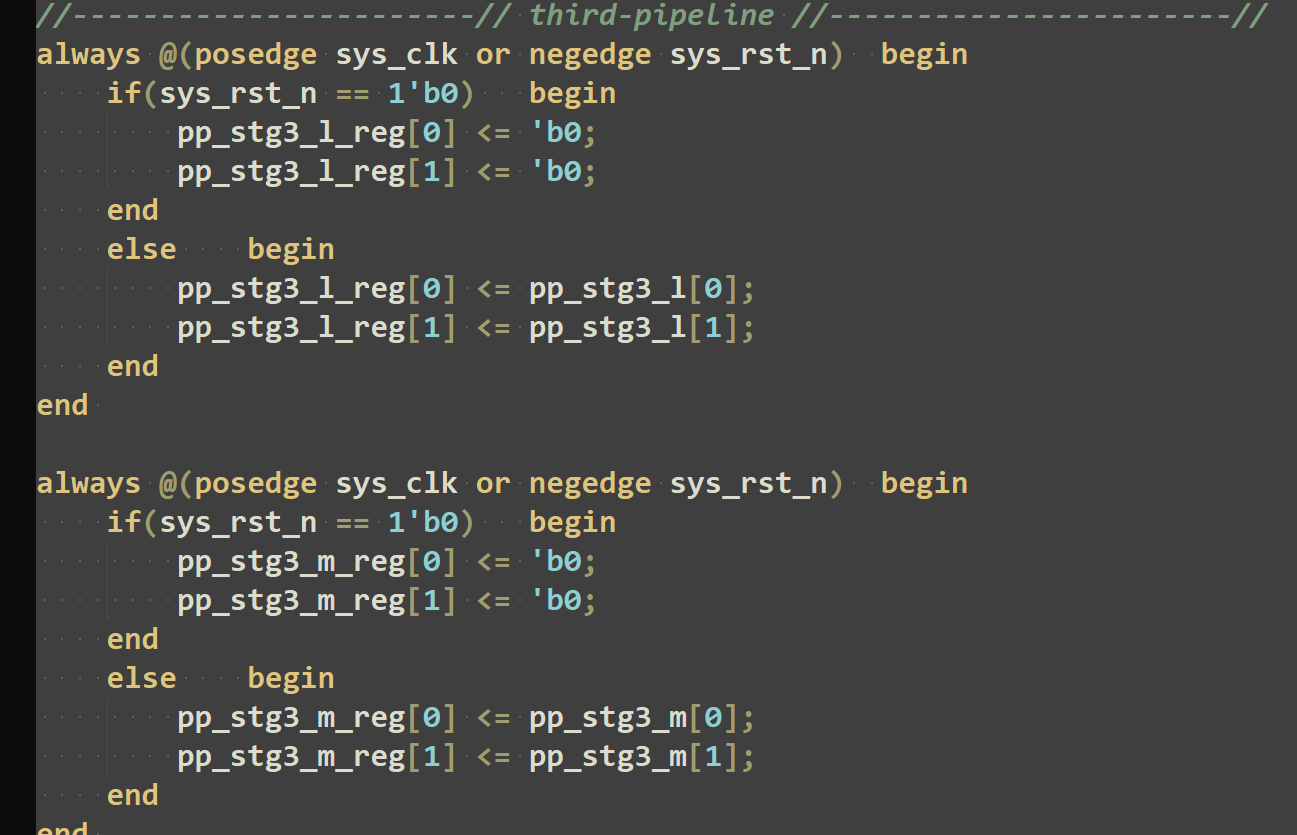
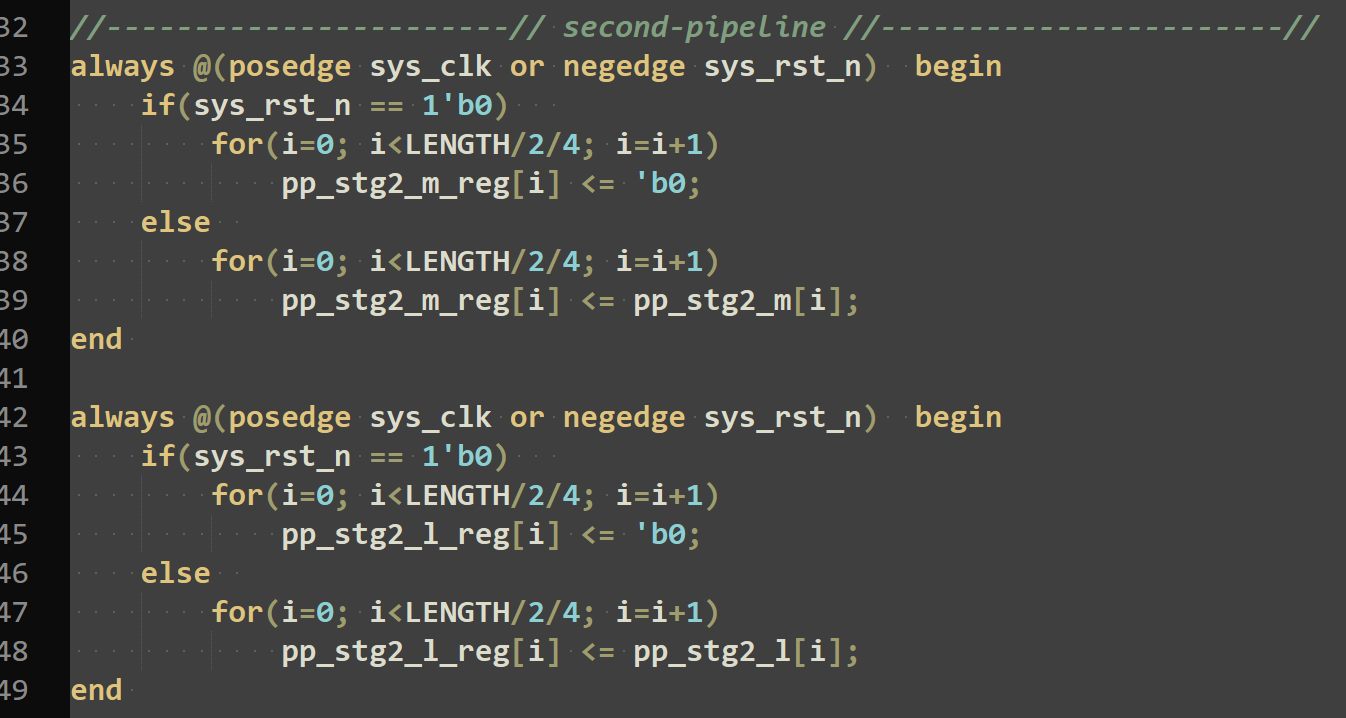
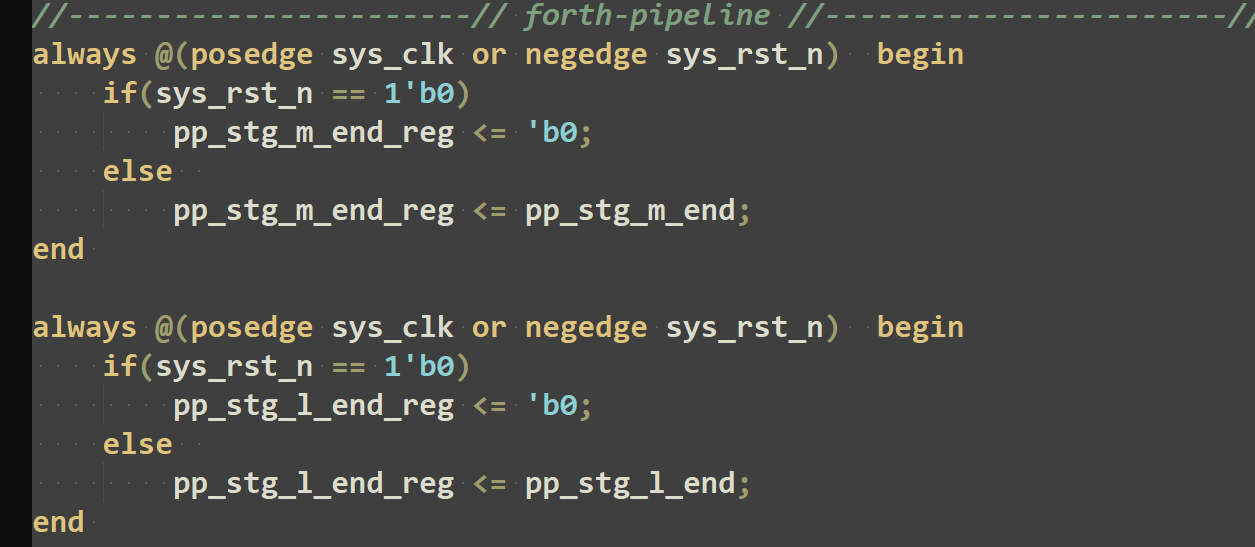
# 二、创新拓展

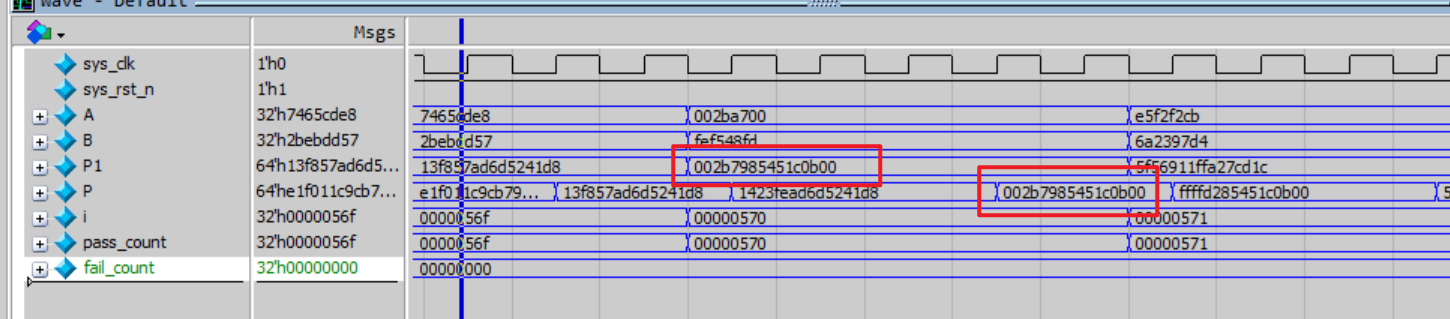
## （一）4级流水的Booth编码器改进

为了增加Booth乘法器性能，提高其吞吐率，现给乘法器加装4级流水。可

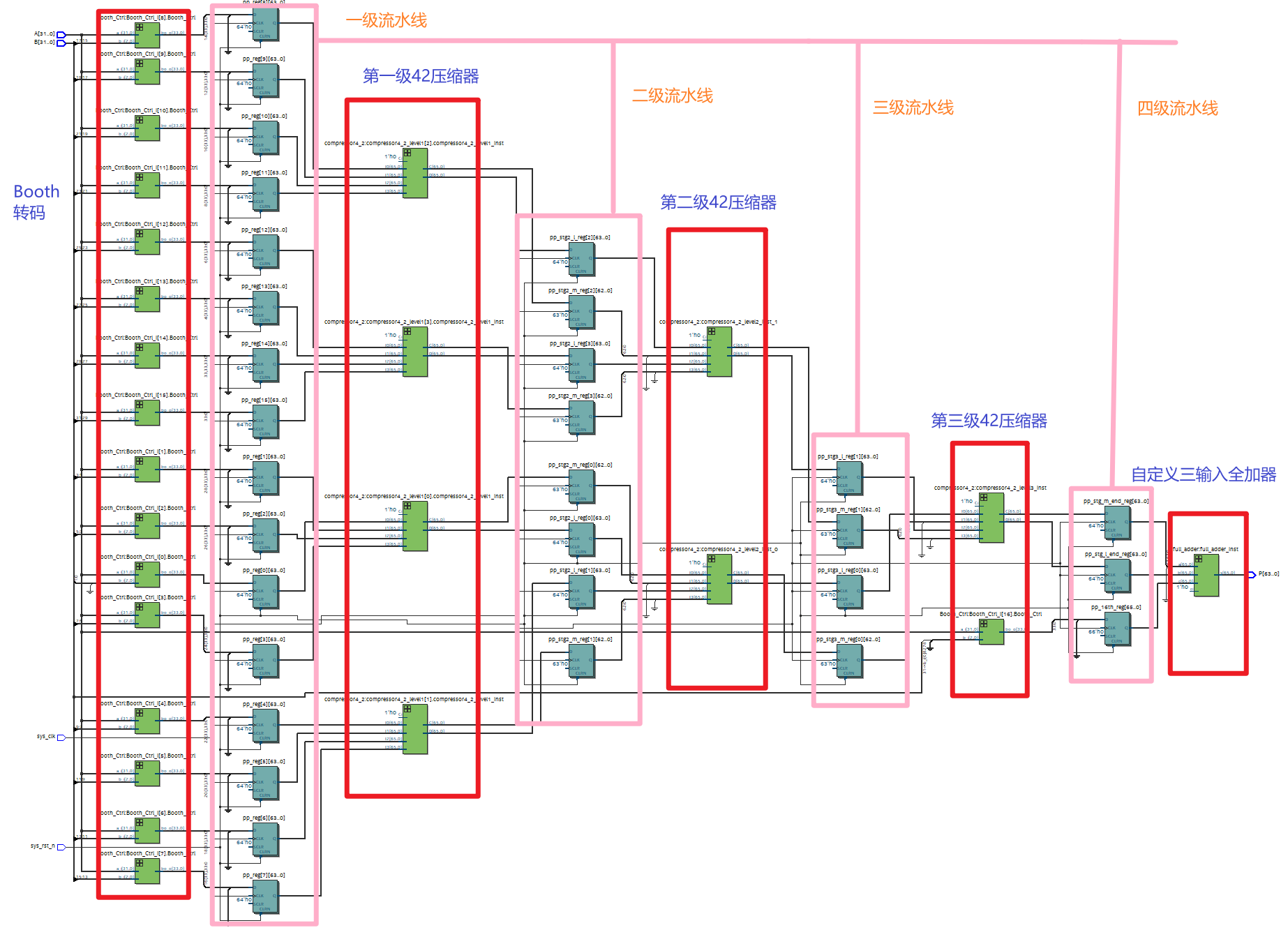
根据实际情况修改确定流水级数





 TB仿真文件修改后，仿真波形如下：

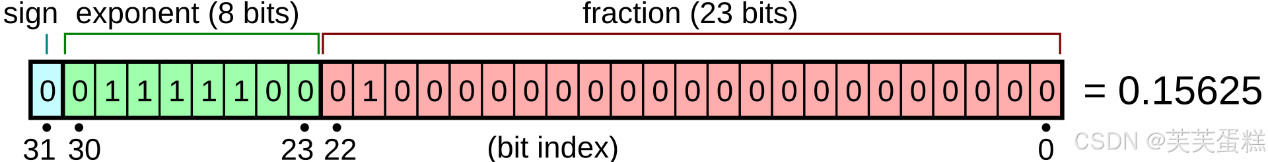
输出结果与软件自带乘法运算对比相同，落后4个时钟周期，仿真成功

RTL视图如下：

Booth乘法器4级流水线改进成功。

## （二）在前面Booth乘法器基础上，设计出2级流水的符合IEEE754标准的32位单精度浮点型乘法器

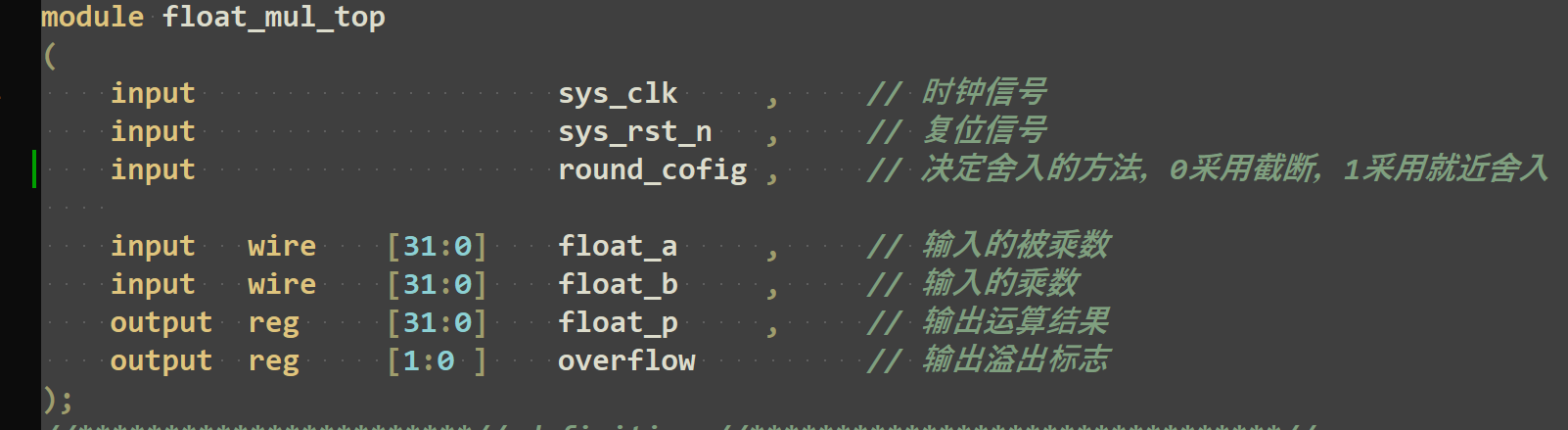
32位单精度浮点型乘法器需要对数字进行解析，最高位为符号位，后面8位为阶码，以2为底，最后23位为浮点数尾数，如图：

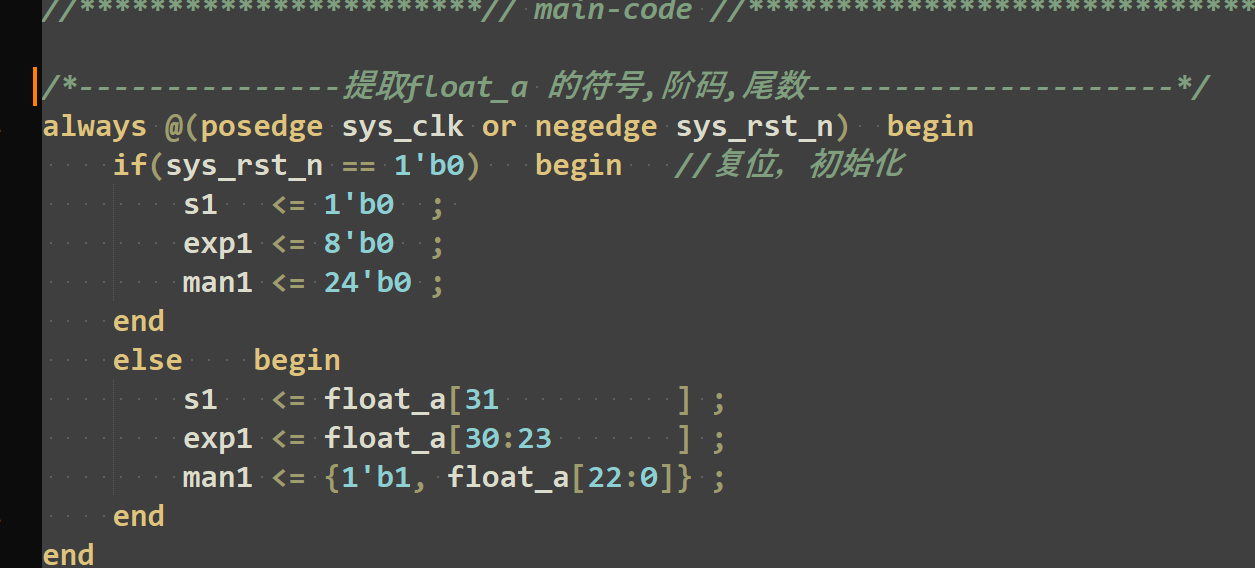
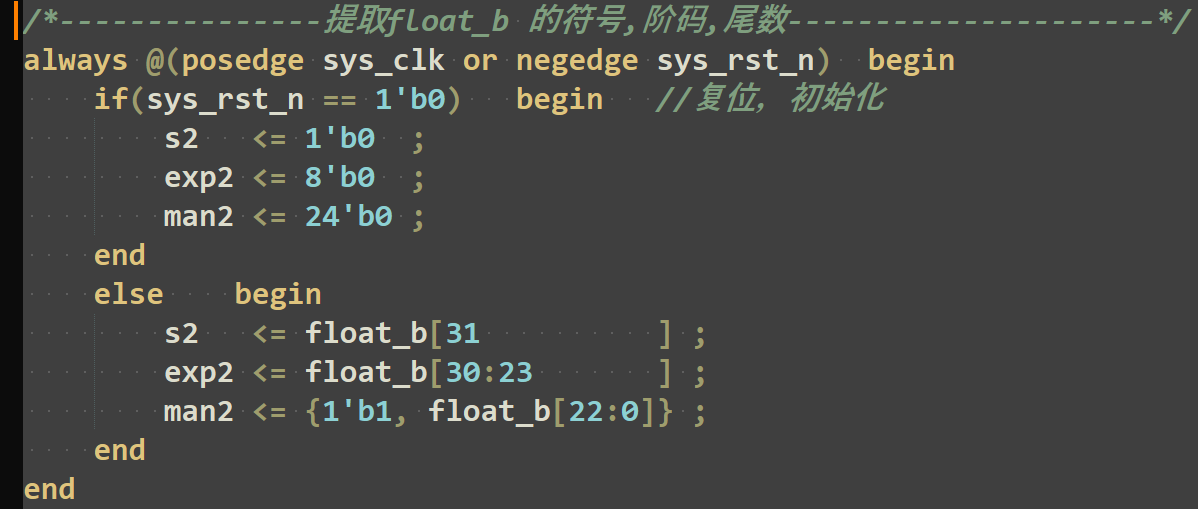


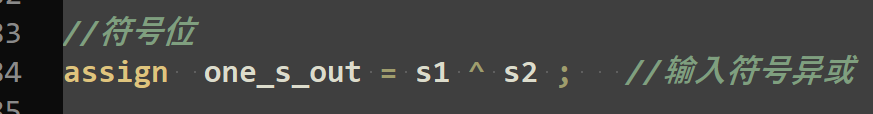
公式为：

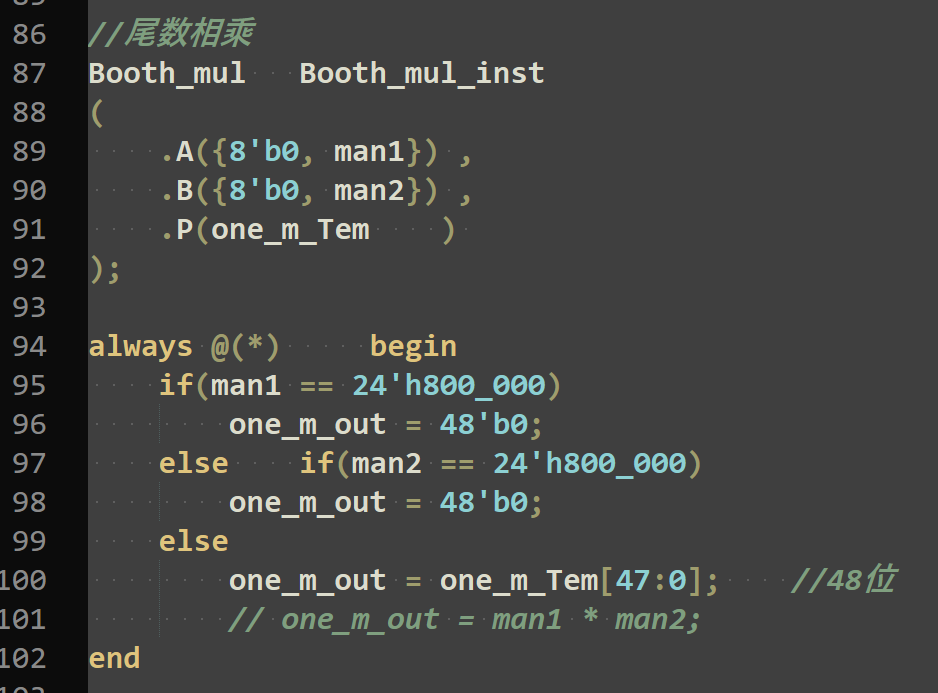


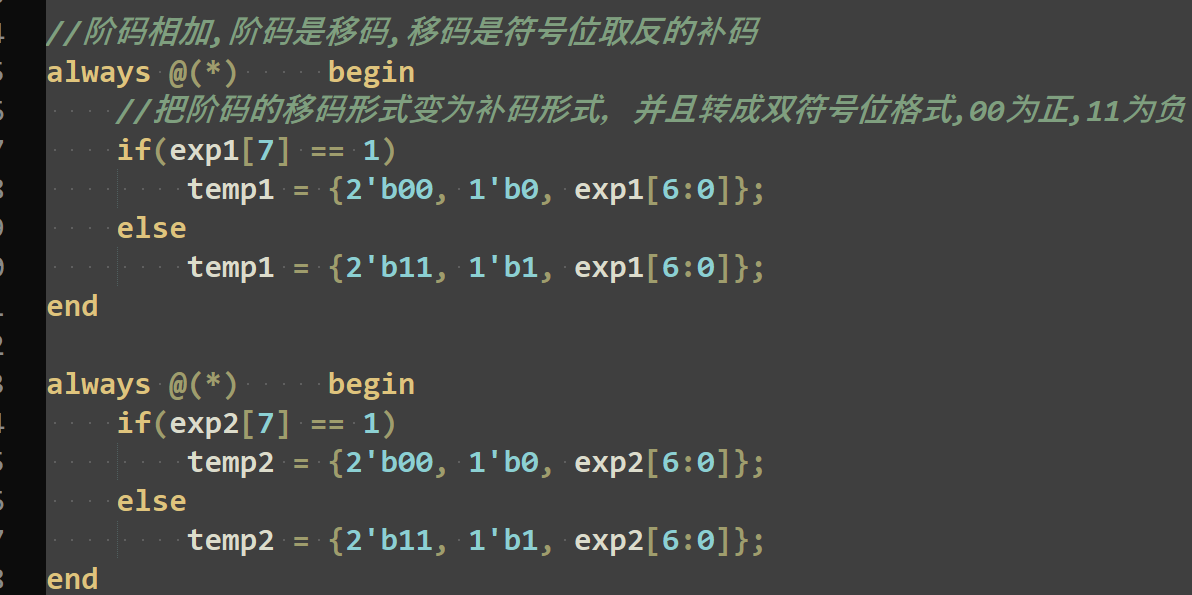
代码：

输入输出端口配置：

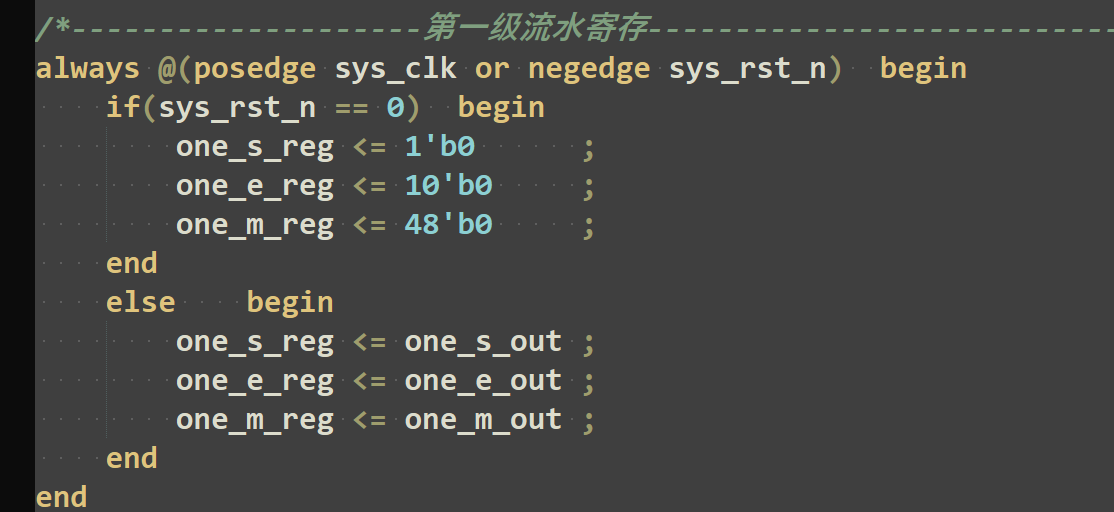
提取乘数float\_a和float\_b的符号位，阶码以及尾数：

判断乘法结果符号：异或判断

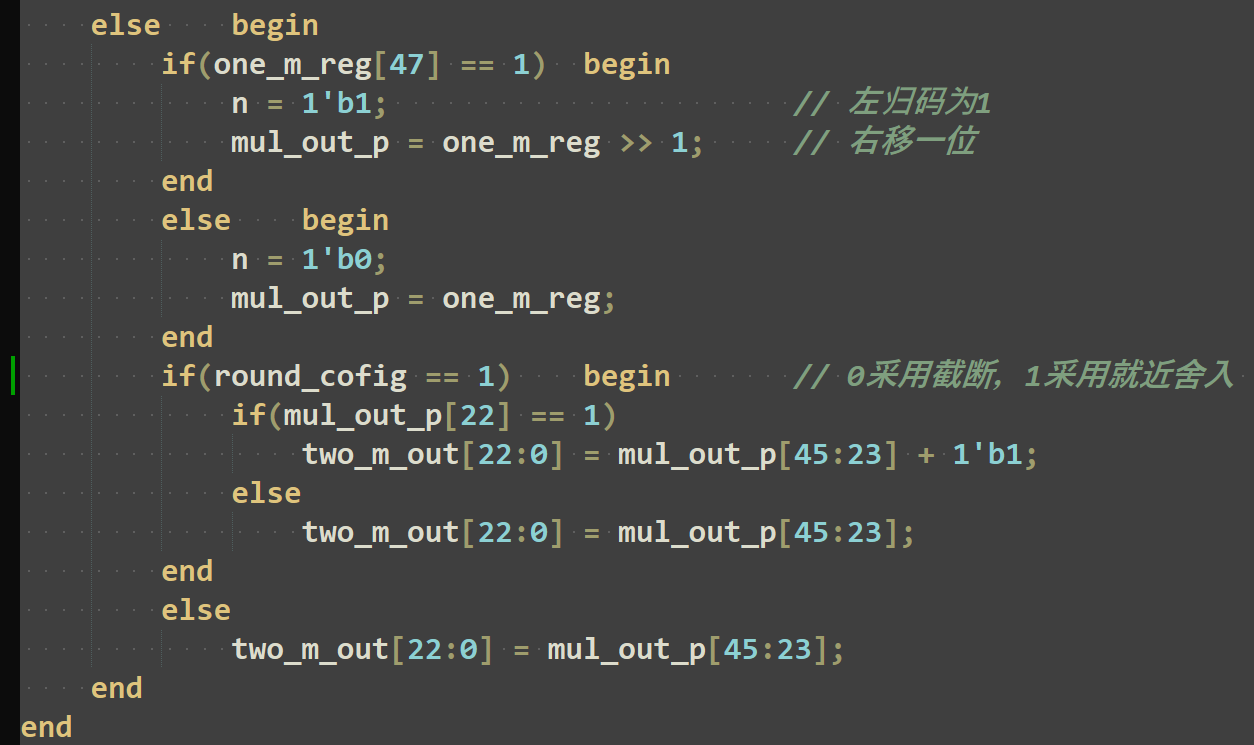
尾数相乘：使用已设计Booth乘法器

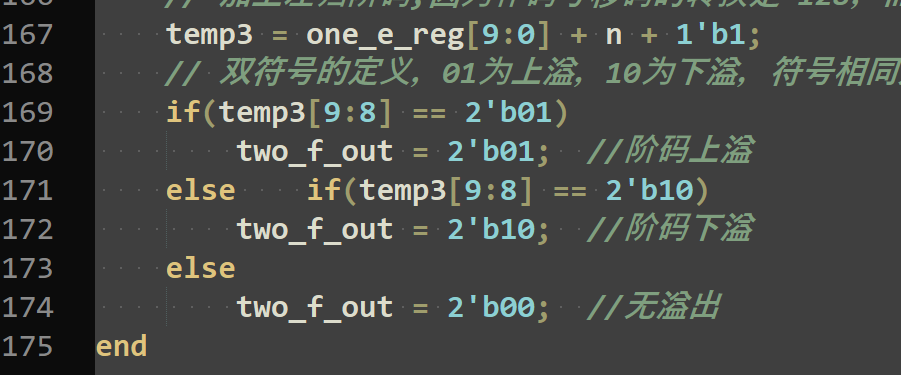
将阶码转成补码形式，并使用双符号位，判断是否溢出

阶码相加：

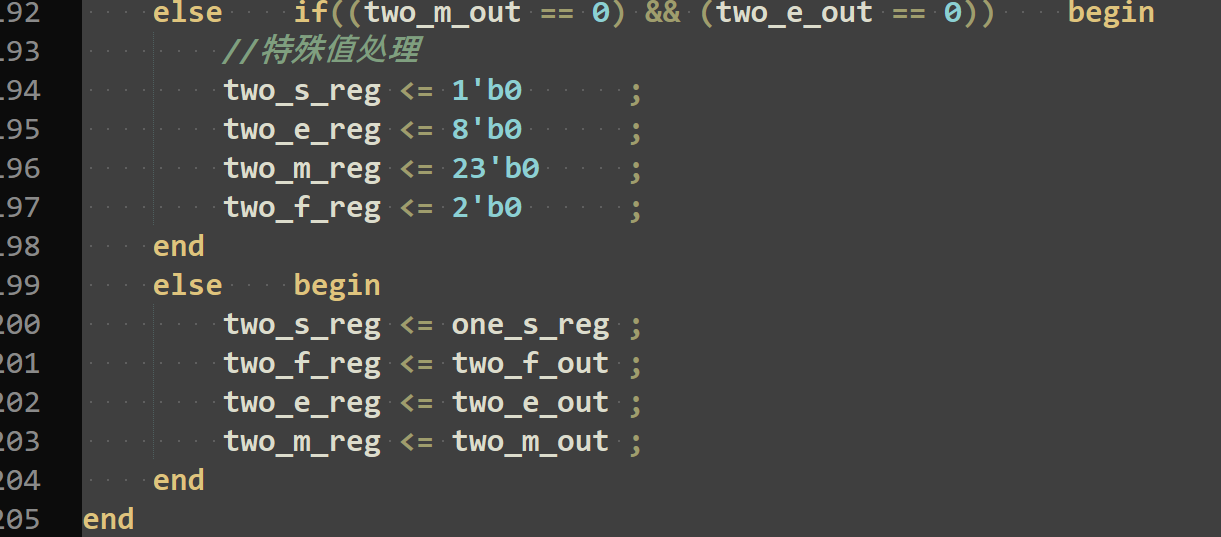
输出的符号位，阶码以及尾数输入第一级流水寄存

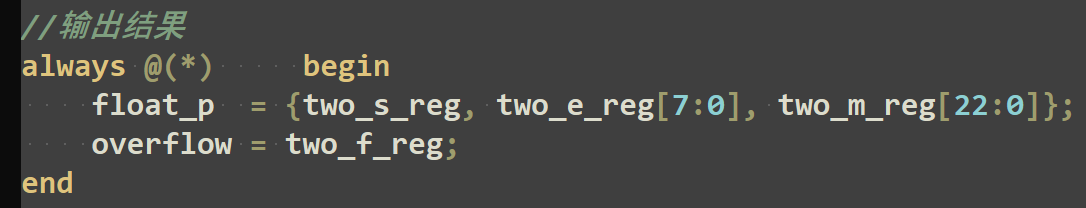
进行尾数规范化，以及舍入处理和溢出判断

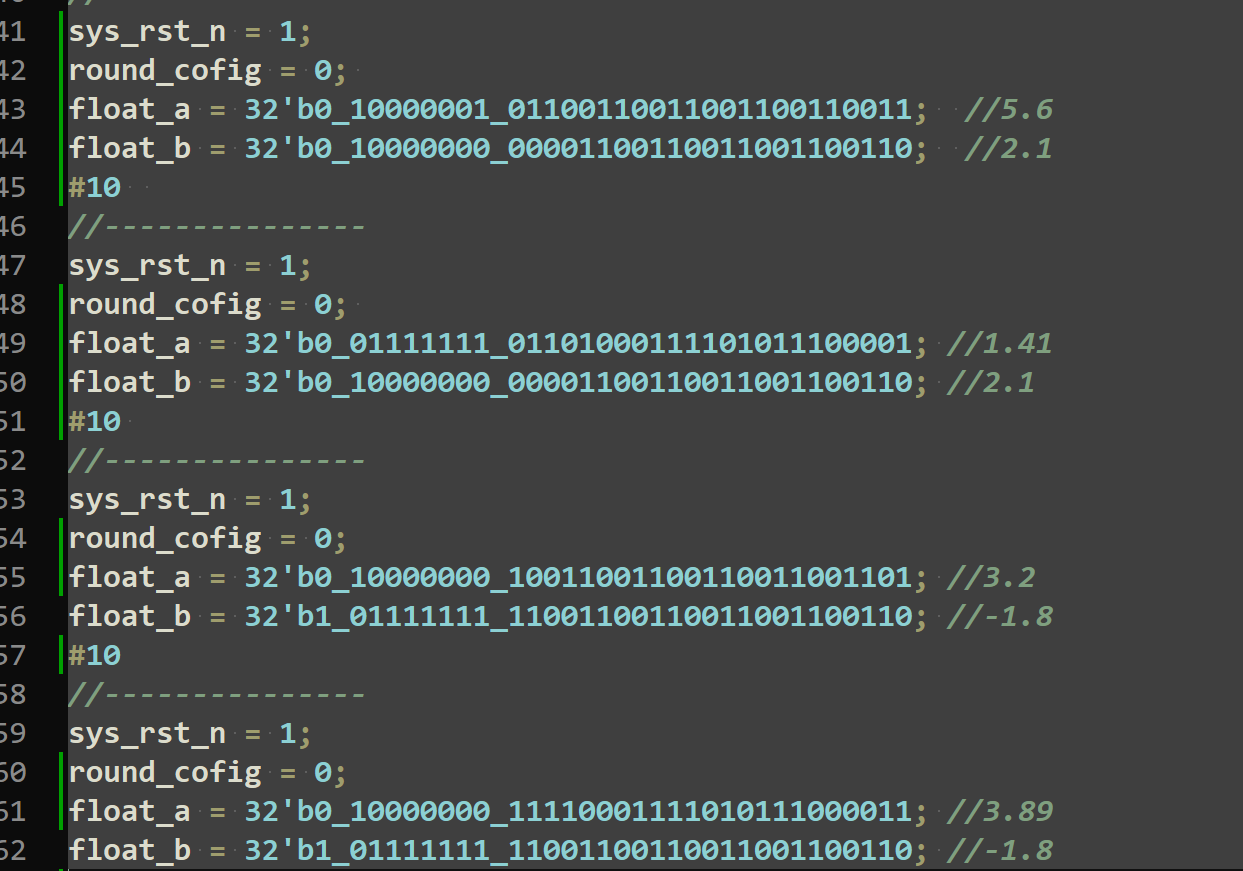


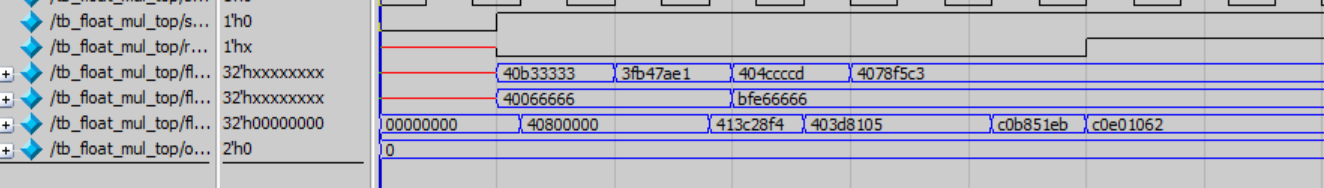


阶码转回移码：

进行第二级流水寄存：

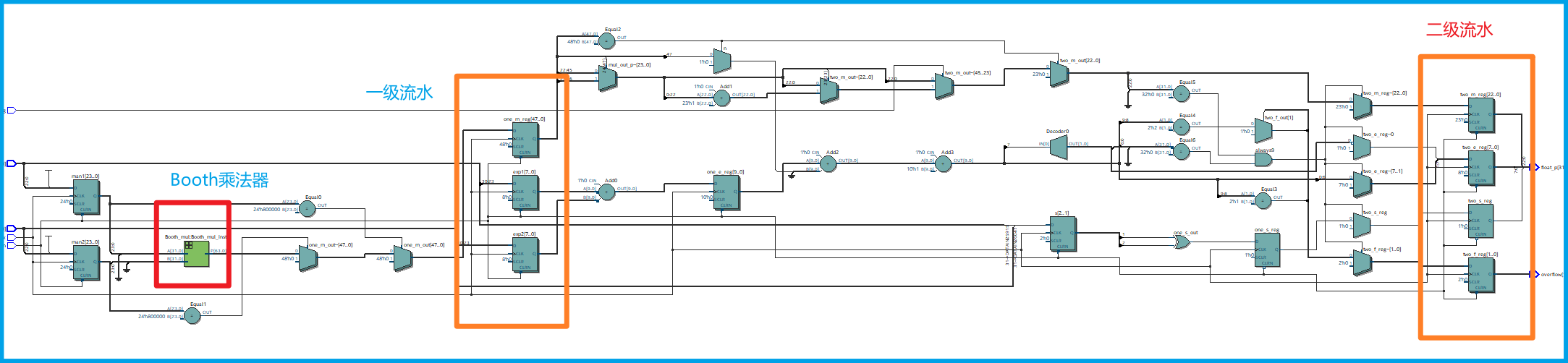
输出结果：

TB文件测试：

输入乘数，modelsim仿真如下：

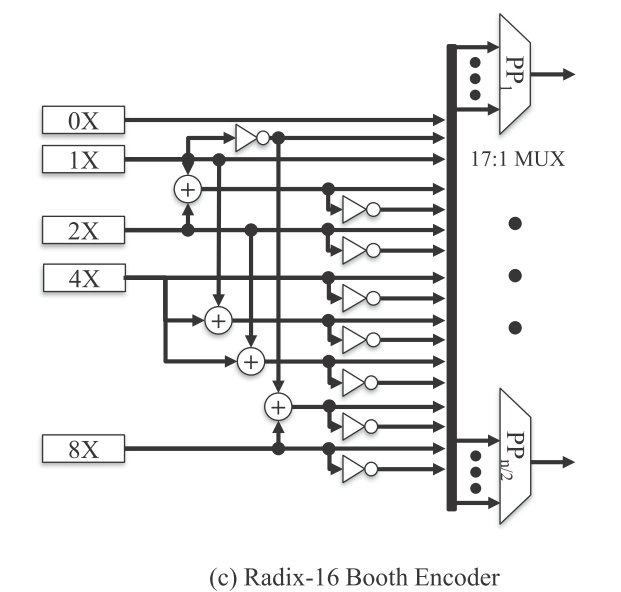
由于2级流水寄存，输出落后输入两个时钟周期。

通过计算，仿真结果正确。

RTL视图如下：

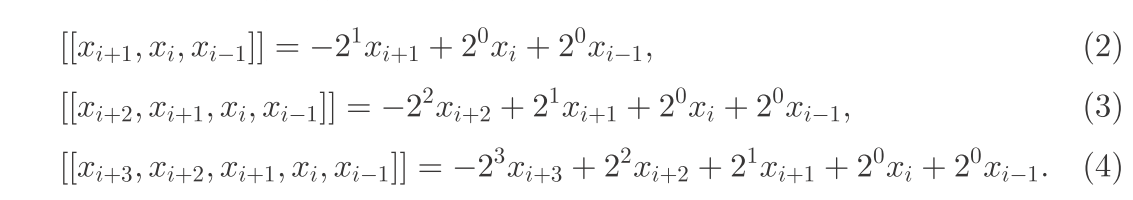
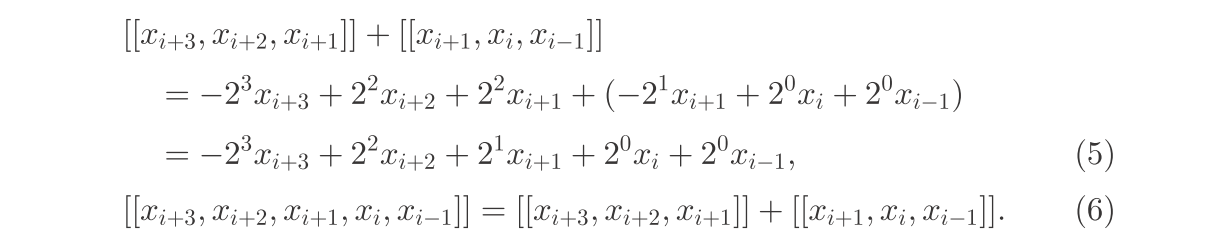
32位单精度浮点型乘法器设计成功。

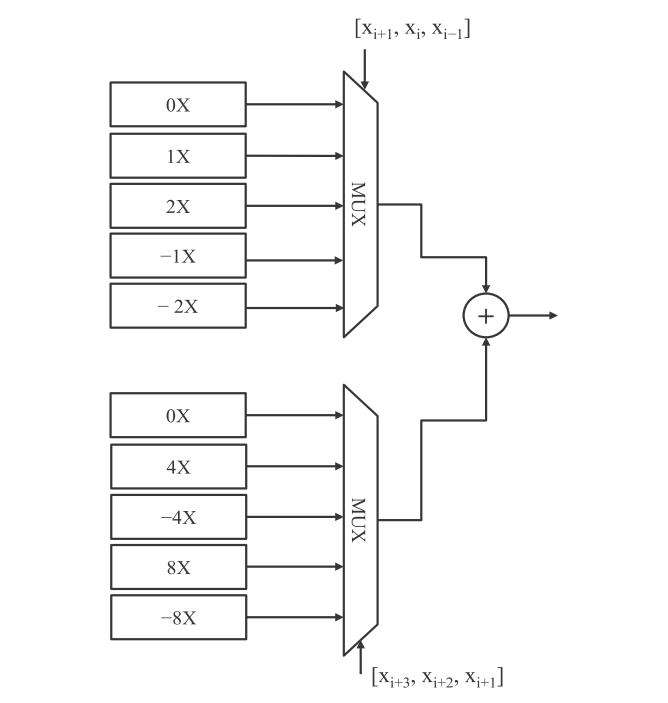
## （三）采用改进的radix-16编码改进radix-4编码乘法器

采用基4布斯编码的乘法相较于传统乘法运算，优化效果已经很明显且易于实现，可以满足大部分应用要求，32位乘法器，甚至64位乘法器都可以采用，是比较常用的一种方式。本次创新采用基16布斯编码，部分积相对于基4又减少一半，加法操作优化效果更加明显。但由于部分积产生逻辑无法单纯通过移位实现，需要引入加法器等其它运算部件，从这方面来看又削弱了优化效果。如果采用基16，需要用以下选择器[1] ：

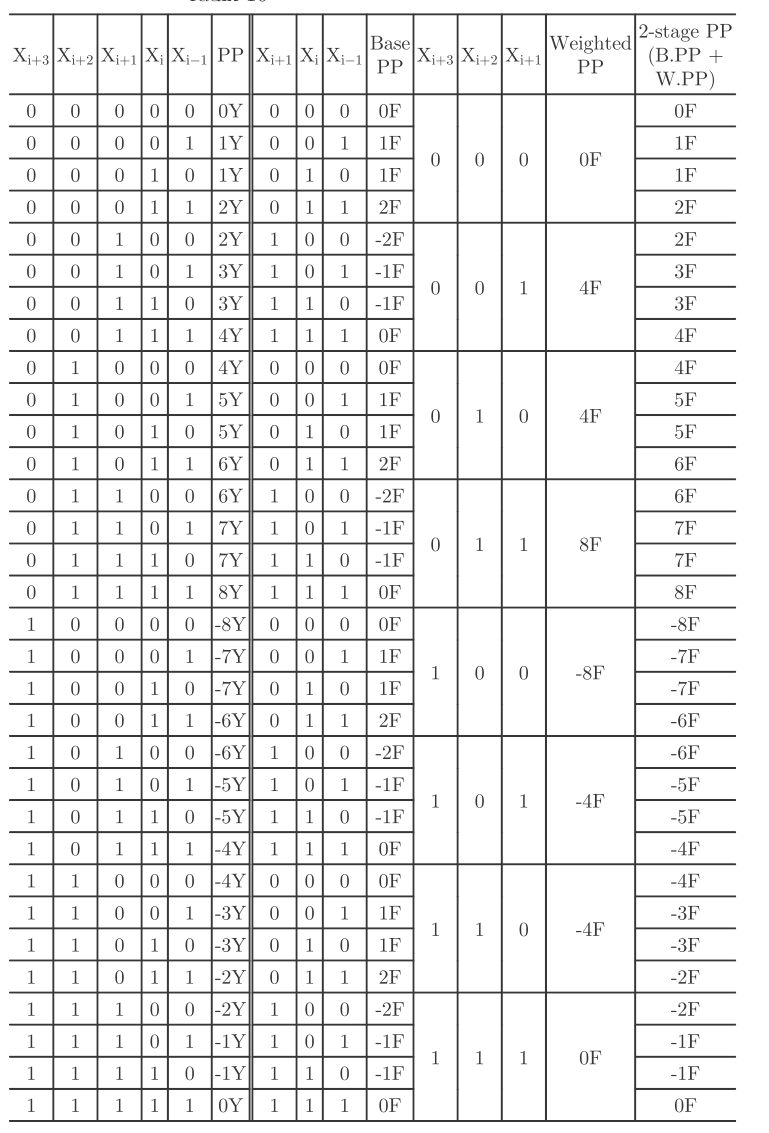
17-to-1选择器会加大实现难度并且所需硬件更多，无法实现优化。

根据算式公式[1]改进：

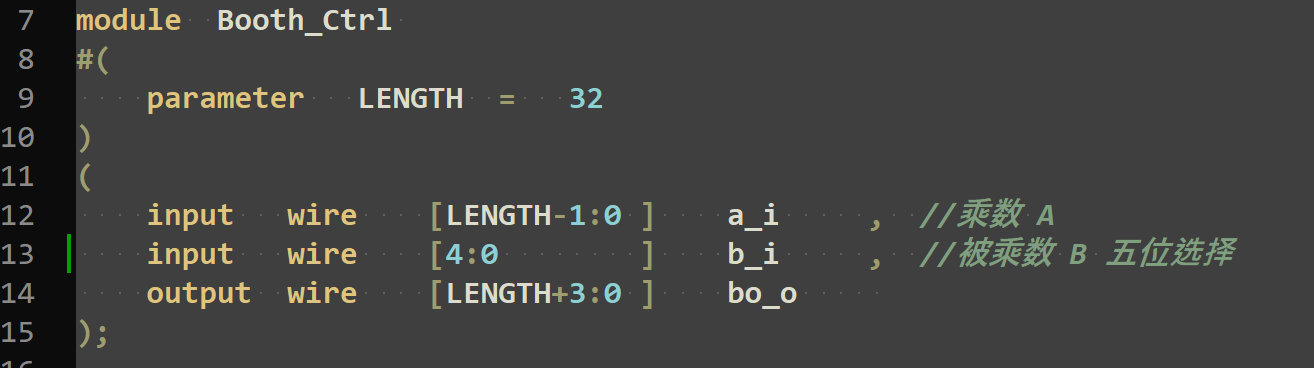
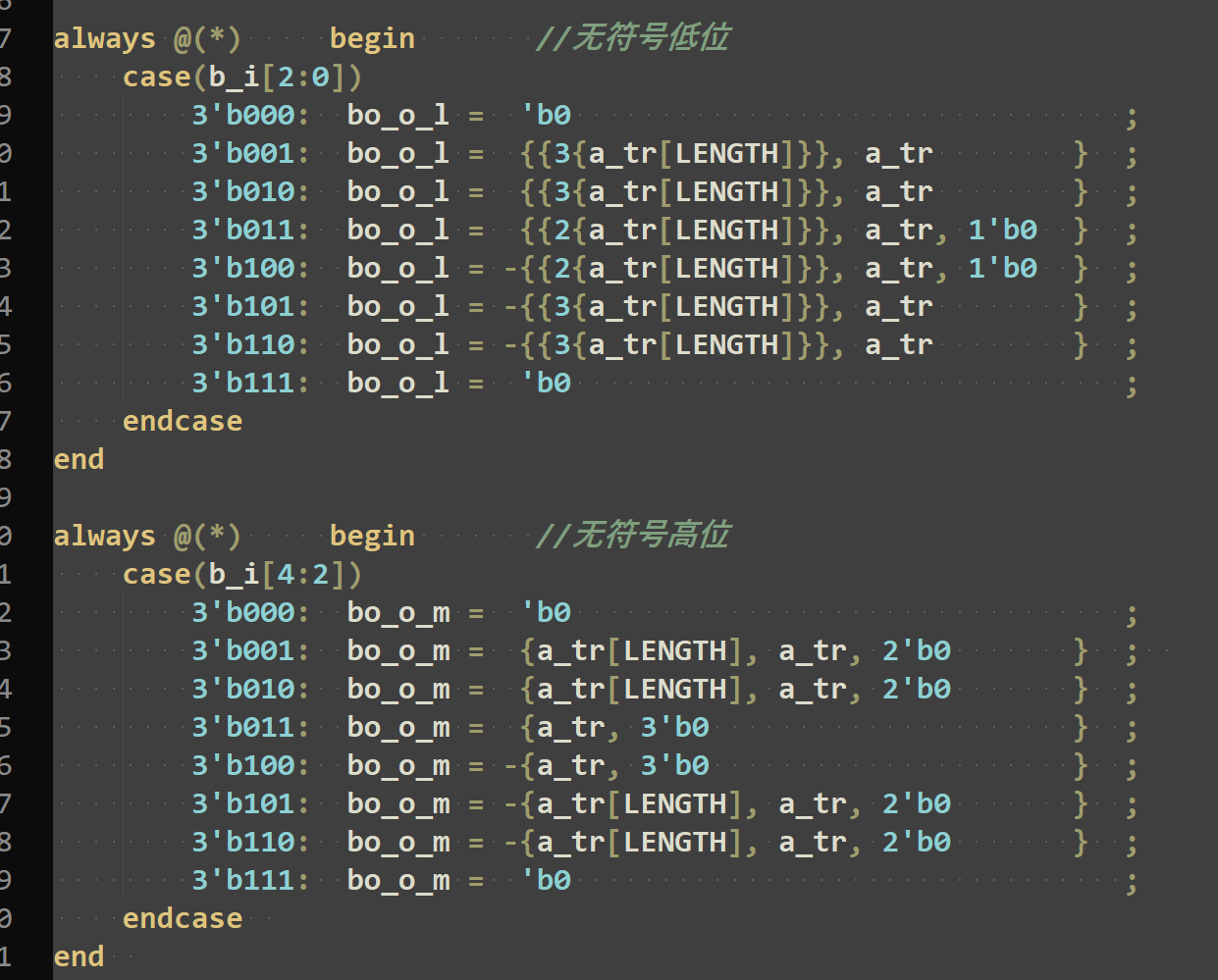
公式（2）是radix-4编码，公式（4）是radix-16编码，而根据公式（5）（6）可轻易得到：基16可由两个基4通过加法操作得到，即，

17-to-1 mux可有2个5-to-1 mux得来：

代码实现：



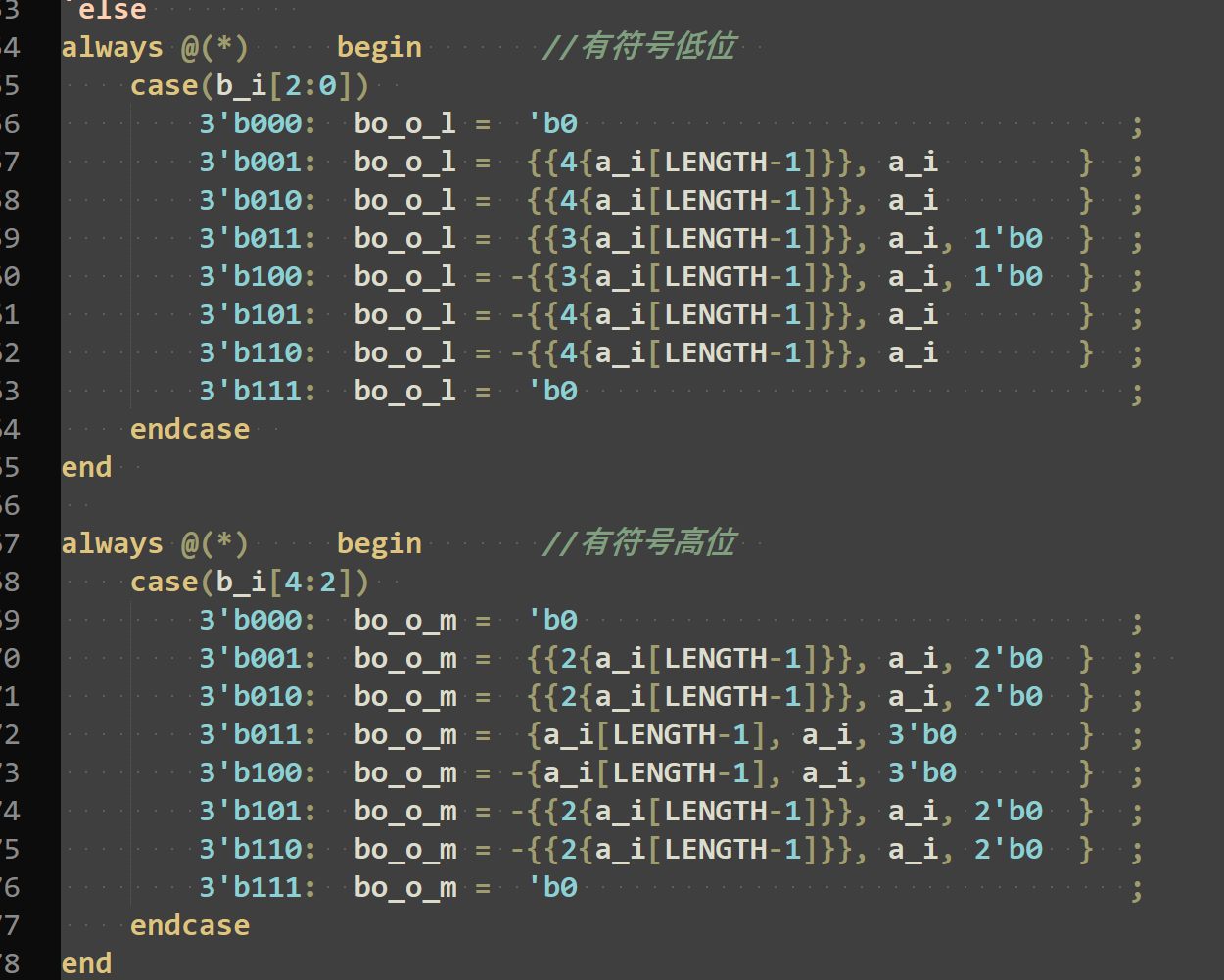
根据查找表实现Booth\_Ctrl模块：

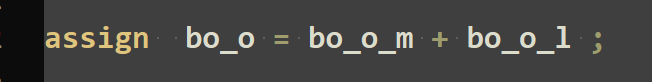
观察到，编码模块输出相对于radix-4位宽多2位

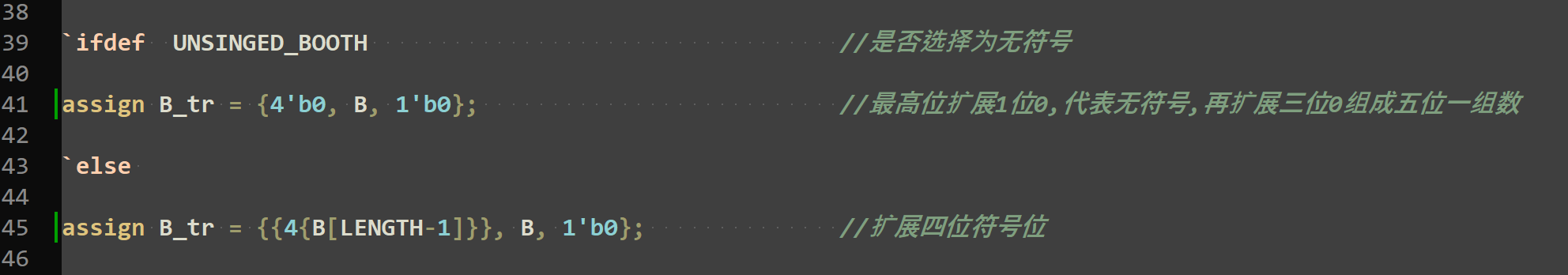
高位编码结果需左移若干位。

无符号实现，使用两个5-to-1mux

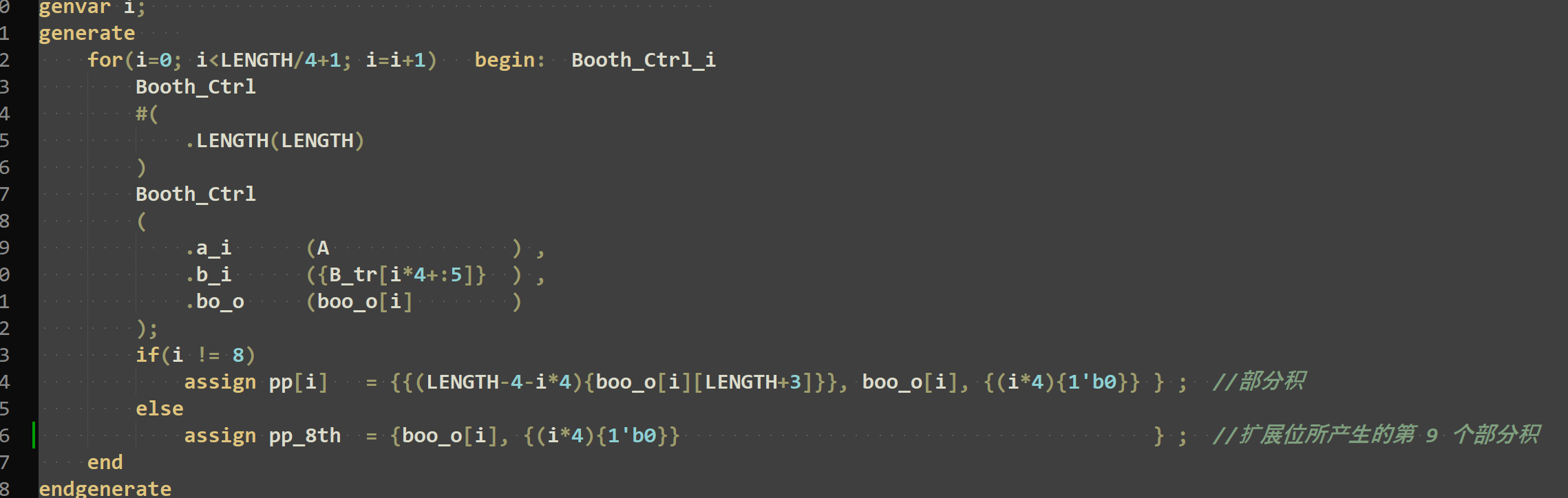
有符号同理，下图：

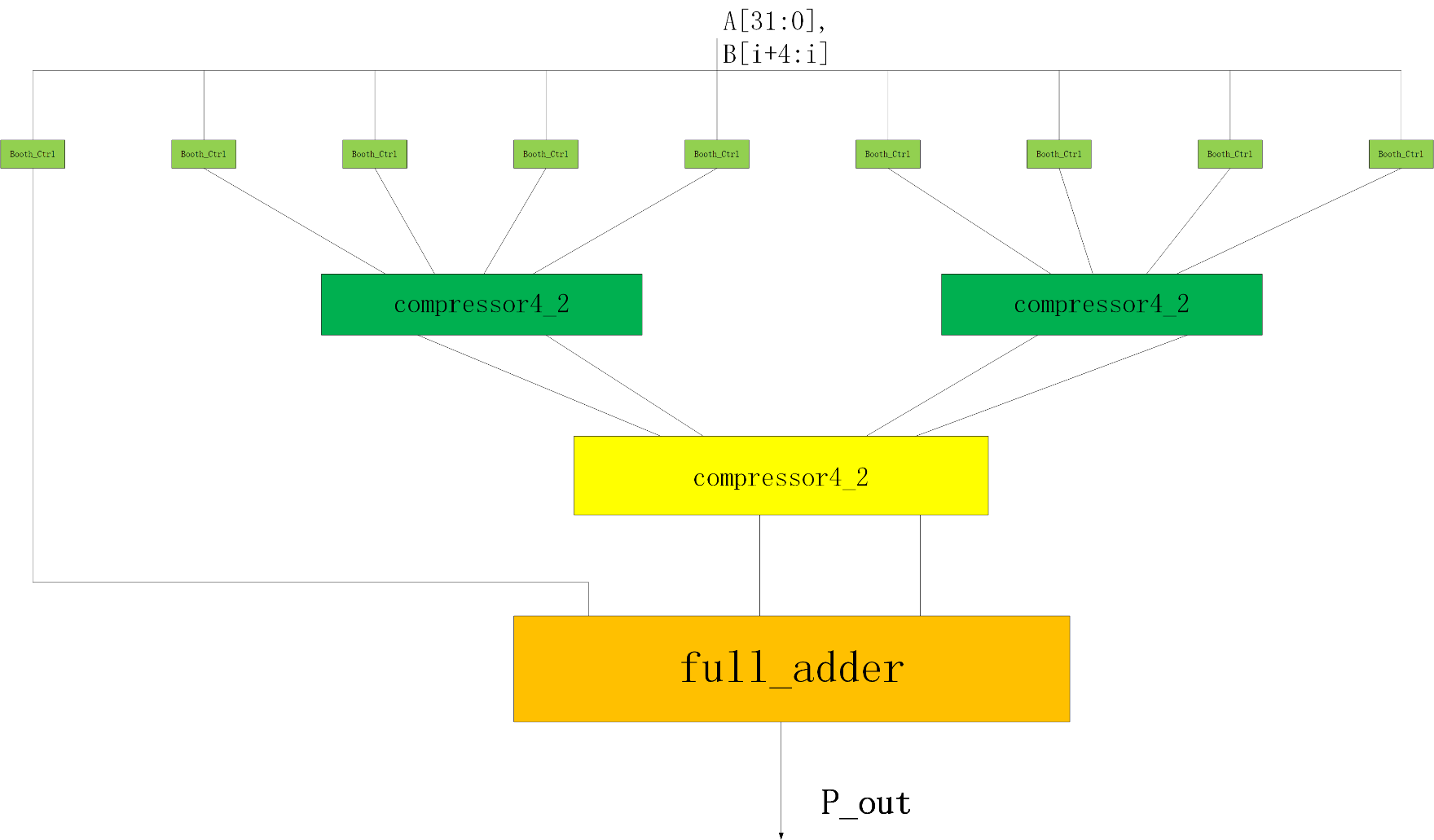


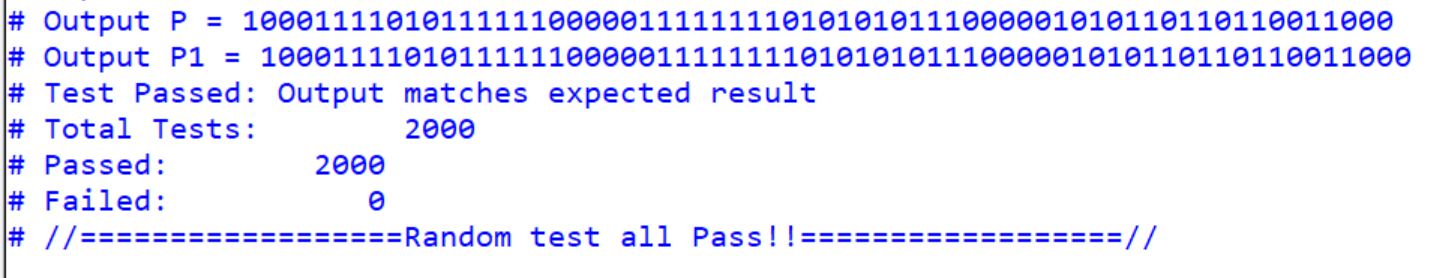
最终通过加法器得到编码输出：

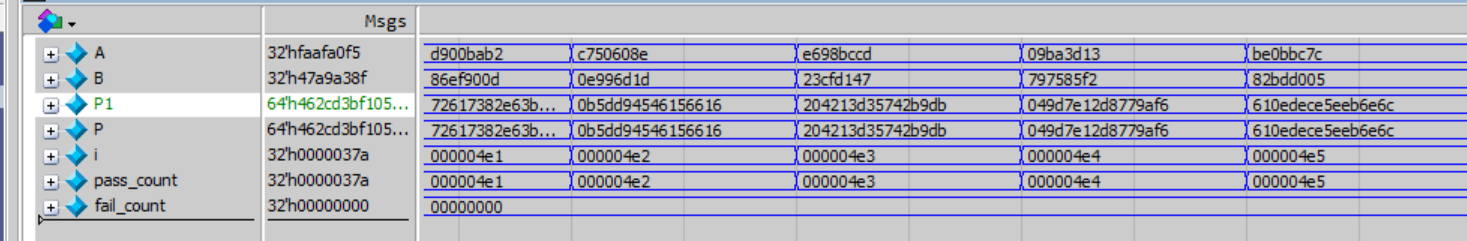
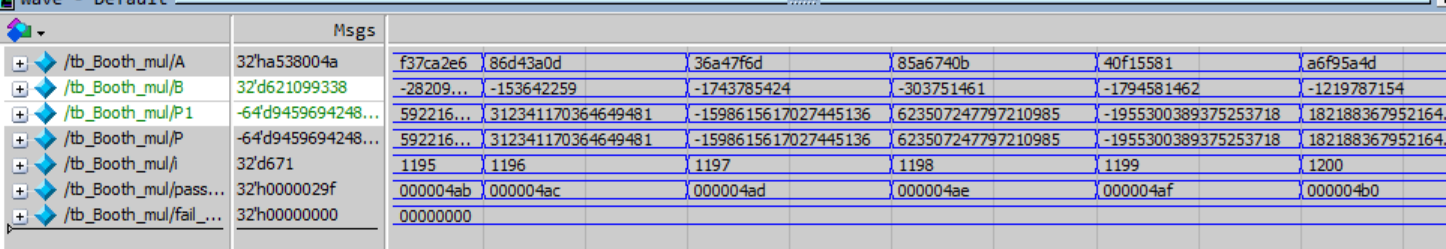
顶层模块改进：

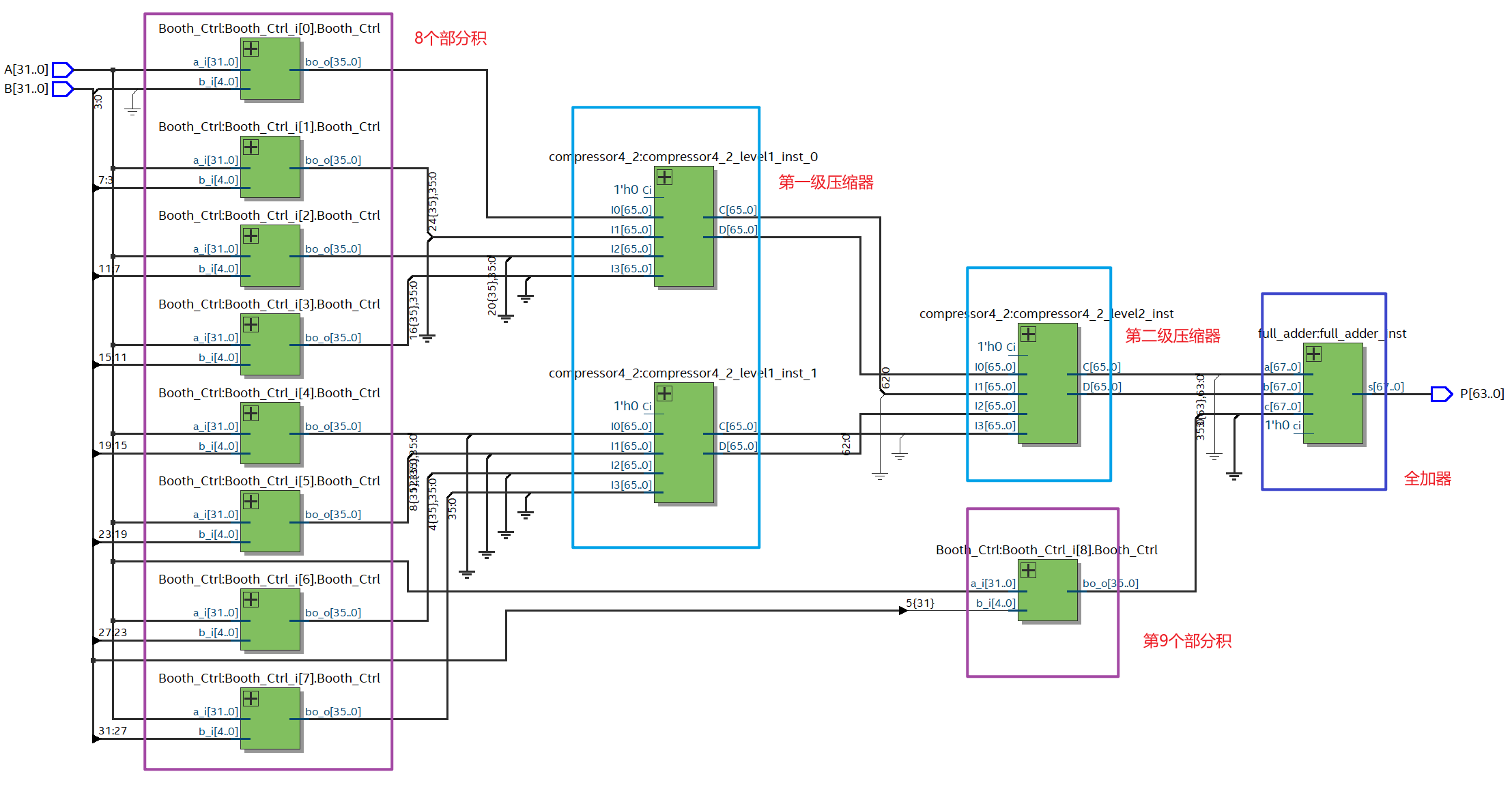
符号位扩展增加

部分积减少为9个，第9个由于扩展而生成。

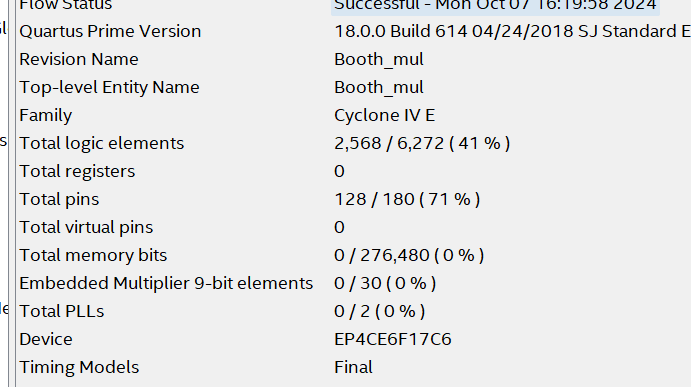
Wallace加法树减少一级，即只用到两级压缩器：

仿真结果如下：

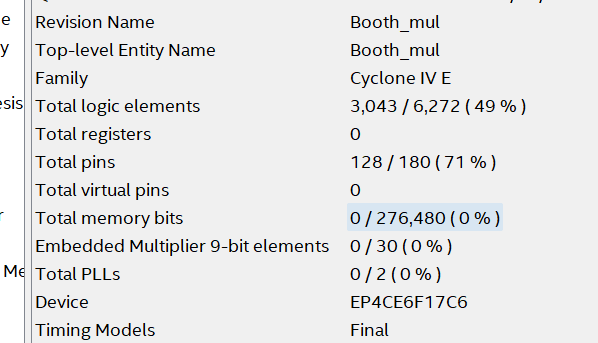
有符号同理：

RTL视图：

仿真成功，改进成功。

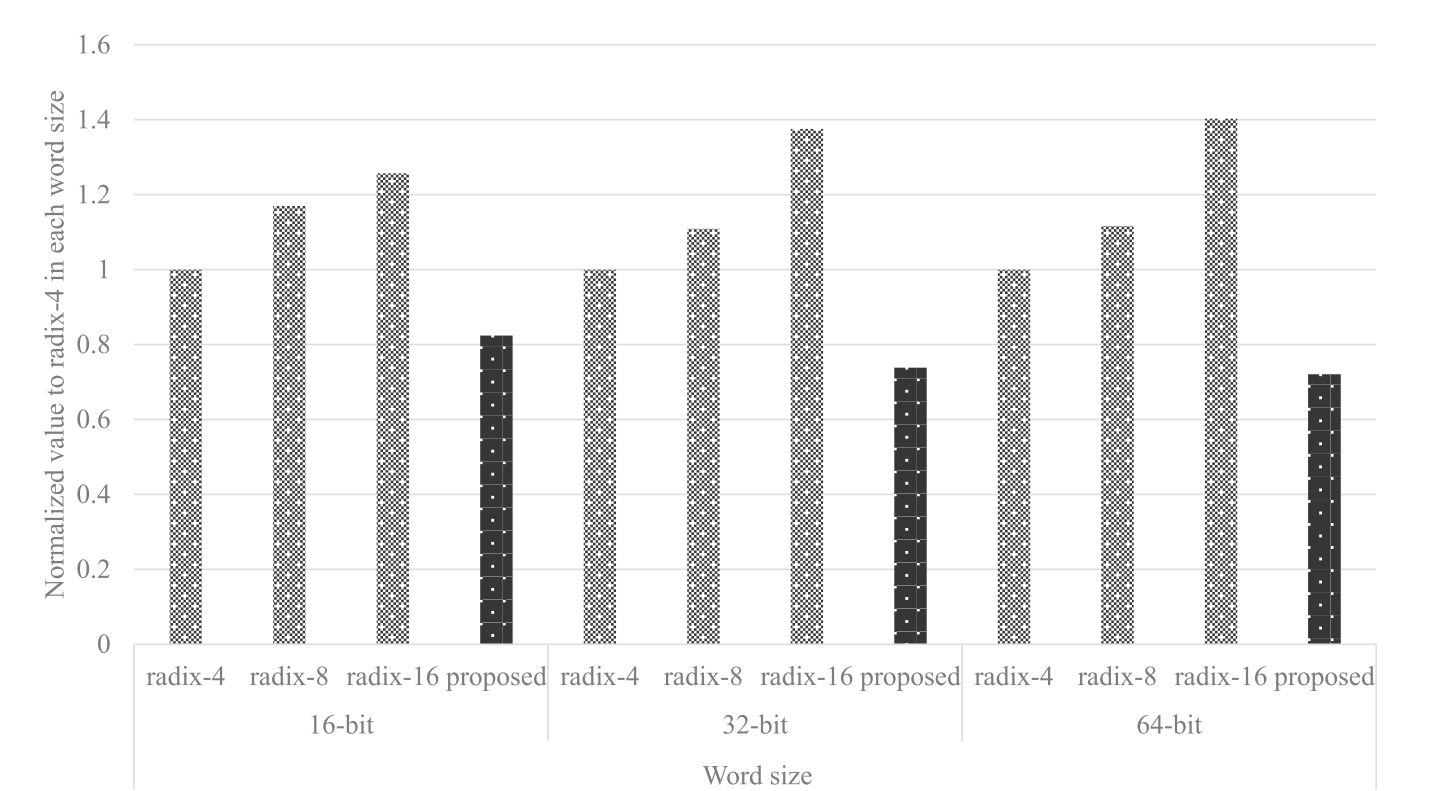
资源消耗对比：

改进的Radix-16



Radix-4

改进的基16逻辑资源明显低于基4.

PD对比：

Column proposed为设计成功。优化效果较radix-4明显

# 三、结论

采用radix-4 Booth编码以及Wallace加法树方法，成功实现设计32位无符号整型快速乘法器，同时可以实现有无符号功能的选择。TB仿真自动测试输出结果对比。并在此基础上，成功实现改进4级流水的Booth编码器，2级流水的符合IEEE754标准的32位单精度浮点型乘法器，与改进的radix-16 Booth编码改进三种创新。

# 参考文献

[1] Ch V S Chaitanya, PSathish Kumar. Design and Analysis of Booth Multiplier with Optimised Power Delay Product[J]. International Conference on Computer Communication and Informatics, Jan. 04 – 06, 2018.

[2] Hyunpil Kim,Sangook Moon,and Yongsurk Lee. Radix-16 Booth multiplierusing novel weighted2-stage Booth algorithm[J]. IEICE Electronics Express,Vol.11,No.13,1-8