实验六 booth 编码乘法器

23342107 徐睿琳

一、实验要求

设计 booth 编码乘法器，了解并掌握 booth 编码的基本原理。

二、实验目的

1、 了解并掌握 booth 编码的基本原理。

2、 学会设计 booth 编码的乘法器。

三、实验内容

1、 设计一个 Radix-4 booth 编码的 4bit 有符号乘法器，设计 booth 编码模块和利用编码信号得到部分积，最终通过加法器相加得到乘积，验证其功能正确。重点在于了解 booth 编码的基本原理和作用及其电路实现。

四、实验步骤

1. Booth 编码乘法原理

Booth 算法通过对乘数进行重新编码，减少部分积的数量，从而提高乘法运算的速度。对于 Radix-4 Booth 编码（基 4 Booth 算法），它将乘数每 3 位一组进行编码，每次处理 2 位，部分积数量减少一半。

设乘数 B 为 N 位二进制补码数 $B = b_{N-1}b_{N-2}\ldots b_1b_0$，并在最低位补 $b_{-1} = 0$。Radix-4 Booth 编码规则如下：

$$PP_i = \text{Operation} \times A \times 2^{2i} \tag{1}$$

其中 A 为被乘数，PP_i 为第 i 个部分积。编码依据 $b_{2i+1}, b_{2i}, b_{2i-1}$ 三位信号决定操作类型。

表 1: Radix-4 Booth 编码表

b_{2i+1}	b_{2i}	b_{2i-1}	操作 (Operation)	部分积生成
0	0	0	0	0
0	0	1	$+1$	$+A$
0	1	0	$+1$	$+A$
0	1	1	$+2$	$+2A$ (左移 1 位)
1	0	0	-2	$-2A$ (取反加 1，左移 1 位)
1	0	1	-1	$-A$ (取反加 1)
1	1	0	-1	$-A$ (取反加 1)
1	1	1	0	0

对于 4-bit 有符号乘法器，乘数 B 扩展为 $b_3b_2b_1b_0b_{-1}$，分为两组：

- 组 0 ($i = 0$): $b_1, b_0, b_{-1} \rightarrow$ 生成 PP_0
- 组 1 ($i = 1$): $b_3, b_2, b_1 \rightarrow$ 生成 PP_1

最终乘积 $P = PP_0 + (PP_1 \ll 2)$。

2. Booth 编码模块设计

设计 Booth 编码模块，根据乘数的 3 位输入生成对应的部分积控制信号。使用组合逻辑电路实现。

3. Booth 选择模块设计

设计 Booth 选择模块，根据 Booth 编码信号选择相应的部分积。使用多路选择器实现。

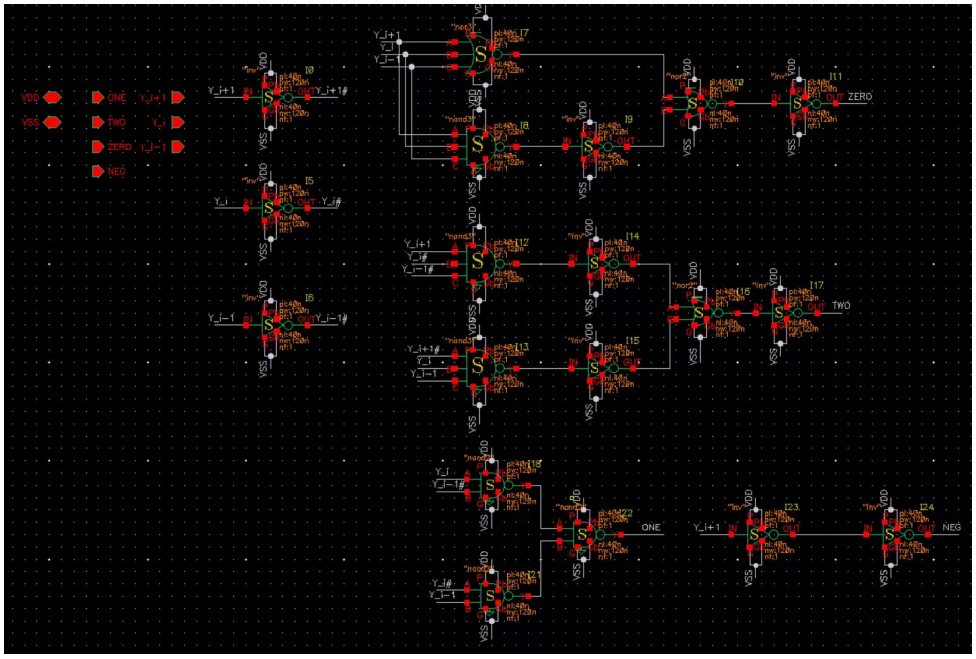

图 1: Booth 编码模块逻辑电路图

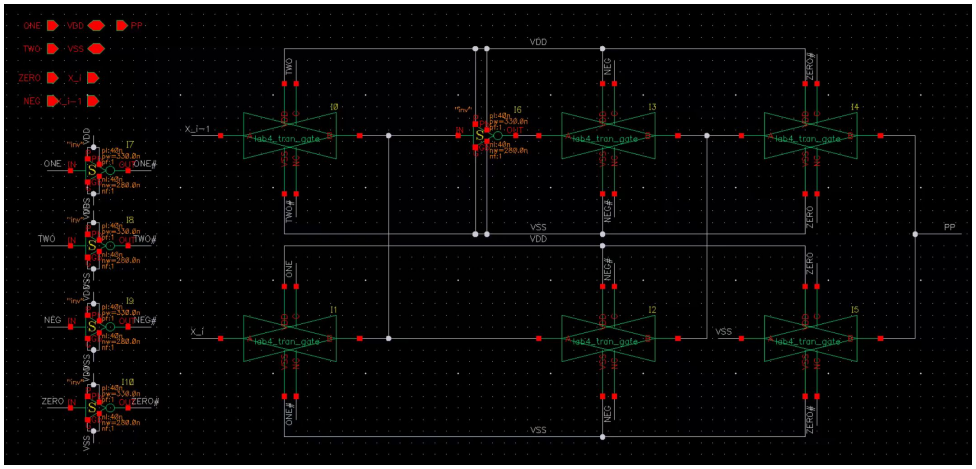

图 2: Booth 选择模块逻辑电路图

4. 部分积加法电路设计

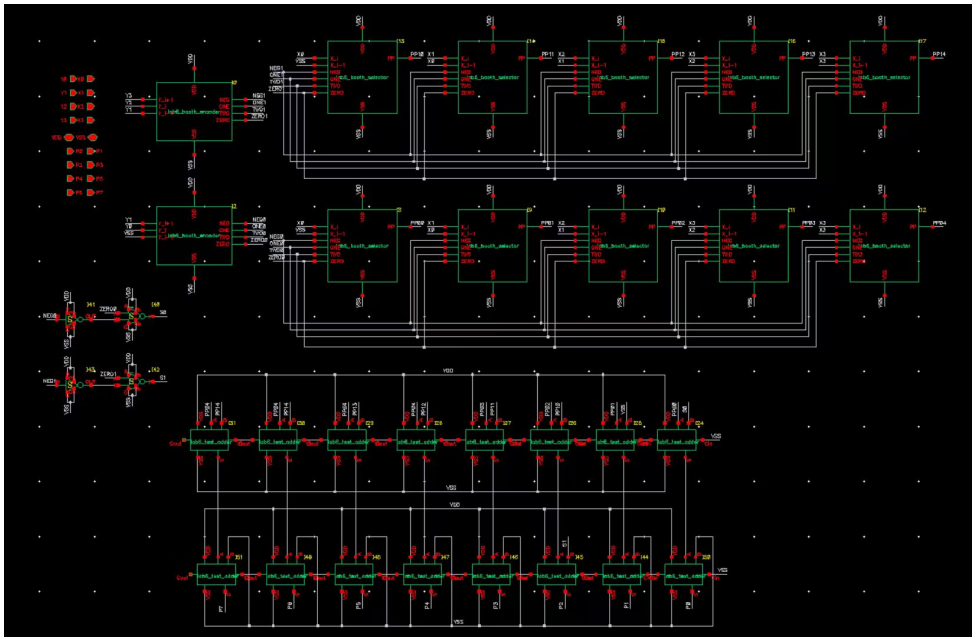

图 3: 顶层乘法模块

设计部分积加法电路，将生成的部分积进行加法运算，得到最终乘积。使用全加器阵列实现。

五、实验结果

如图所示，仿真结果验证了设计的 Radix-4 Booth 编码乘法器的正确性。输入被乘数和乘数后，输出乘积符合预期结果。

六、总结

通过本次实验，我深入理解了 Radix-4 Booth 编码乘法器的工作原理及其在数字集成电路设计中的重要性。

首先，在理论层面，我掌握了 Booth 算法的核心思想：通过对乘数进行重新编码（Recoding），将连续的“1”序列转换为“+1”和“-1”的操作，从而减少部分积（Partial Products）的数量。对于 Radix-4 Booth 算法，每两位乘数生成一个部分积，使得部分积的数量减少了一半（对于 N 位乘法器，部分积从 N 个减少到 N/2 个）。这显著降低

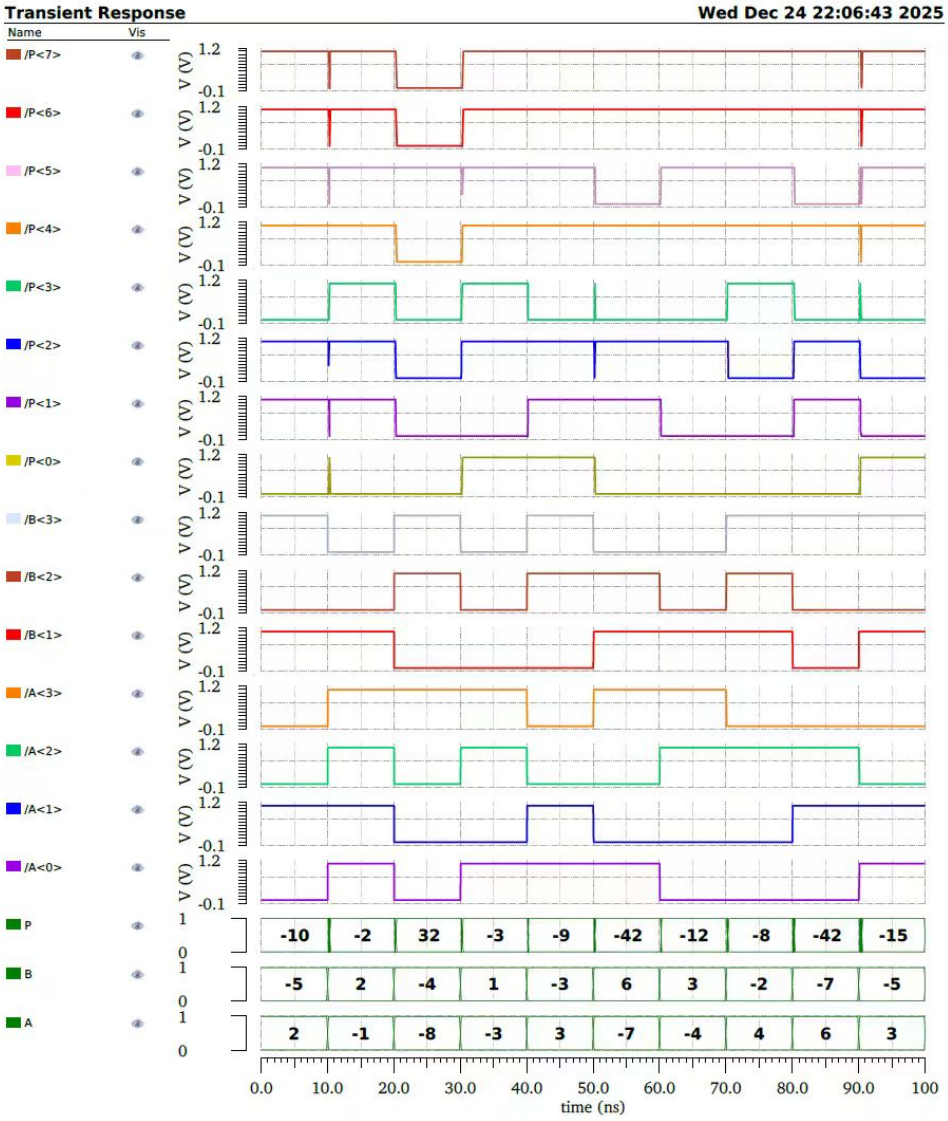

图 4: Booth 编码乘法器仿真波形图

了后续加法树（如 Wallace 树或压缩器阵列）的复杂度，从而提高了乘法器的运算速度并减少了面积和功耗。

其次，在电路设计层面，我学会了如何将算法转化为具体的硬件模块：

- **编码模块（Booth Encoder）：**实现了根据相邻三位乘数生成控制信号（如单倍、双倍、取反等）的逻辑。
- **部分积生成模块（Partial Product Generator）：**利用编码信号和被乘数生成实际的部分积，特别是处理负数时的“取反加一”操作（补码表示）以及移位操作。
- **加法阵列：**理解了符号位扩展（Sign Extension）在部分积相加时的重要性，确保了有符号数运算的正确性。

最后，通过仿真验证，我确认了设计的 4-bit Booth 乘法器能够正确处理正数、负数以及零的乘法运算。本次实验不仅锻炼了我的逻辑电路设计能力，也让我体会到了算法优化对硬件性能提升的关键作用，为后续设计更高位宽、更高性能的运算单元打下了坚实基础。

附录

附录 A 相关代码

附录 B 数据表格

附录 C 其他材料