计算机系统贯通课程-计算机系统 I

——Lab2:64 位全加减法器

洪奕迅 3230102930@zju.edu.cn

史璐欣 3220104390@zju.edu.cn

计算机学院
浙江大学

2025 年 3 月 20 日

新的语法

大家已经学习了 `wire,reg,module,assign,always,initial` 这些基本的语法，掌握这些语法已经足够大家写出很多模块和必需的仿真．但众所周知，懒是生产力发展的重要驱动力，可以考虑以下开发中可能出现的问题：

- 你需要例化一大堆被选择数据位数各不相同但路数相同的复合选择器
- 在你的代码中反复重复一条/几条语句（e.g. max/min in C）
- 你希望遍历一组数据/对一组数据执行某一类相同操作但你懒得手动遍历
- 你想快速生成能覆盖大部分情况的测试样例

这些场景分别对应接下来讲的 `parameter,function,for`，`random`，也都会在这次 Lab 中得到应用．（Reference：ChipVerify）

Parameter

以我们要实现的加法器为例，实际需求中我们可能会同时需要不同位数的加法器，但其原理完全一致，而只是为了位数差异去进行重复封装显然是麻烦且抽象的. 因此为了提高开发的灵活性，我们强烈建议大家在实现一些可能复用器件时使用参数式. 我们主要会使用的参数类型包括 `oarameter`,`localparam`:

- `parameter`: 类似于 C 中的函数的形参，在部件被例化时传入，如果不声明其值则会保持模块被定义时设定的初值（现在你可以看懂 logisim 生成代码中的 `BubbleMask` 了）
- `localparam`: 类似于 C 中的 const 常量，在被例化模块的整个生命周期中保持不变
- 你可以使用任何参数和立即数等给参数赋值，但不能使用电路输出为其赋值
- `` `define ``: 如同你在 C 程被坑的每一次，这个 `define 也只做文本替换的活，而且大家也不怎么会用到

function

- 包含多个输入和一个输出，且这个输出就是以函数名隐式声明的变量
- 同 `module` 一样，声明的变量作用于函数内部
- 不能包含任何时序成分
- 不能包括 `assign` 和非阻塞赋值

for/generate for

- `for`：和 C 的循环用法基本相同
- `generate for`：一般用来同时例化多个模块（在其他场景中也可以使用，和普通的 `for` 区别不大，但需要注意两者用法带来的细微差异），需要注意循环变量需要是 `genvar` 类型

$random()

- 随机数生成器，且只能用于 intial 块内
- 可以添加 seed 参数，不添加则每次随机，添加后则随机数取决于 seed

有了 $random() 和 for，就可以快速生成测试样例了.

加法器

在小学数学我们就学过，对于多位数加法，一个比较好的方式是列竖式．而竖式的核心就是通过逐位加法来降低运算的复杂度，再通过进位来建立不同位的联系．我们实现的加法器也是如此，只不过运算者从人变成了机器，因此数字变成了二进制．
但这显然是更简单的，在不考虑进位的情况下，我们会发现二进制加法其实和异或完全等价，而异或门实现代价是很低的，因此对于怎么进行两数每一位的加法我们已经有了答案:

$$s = a \oplus b$$

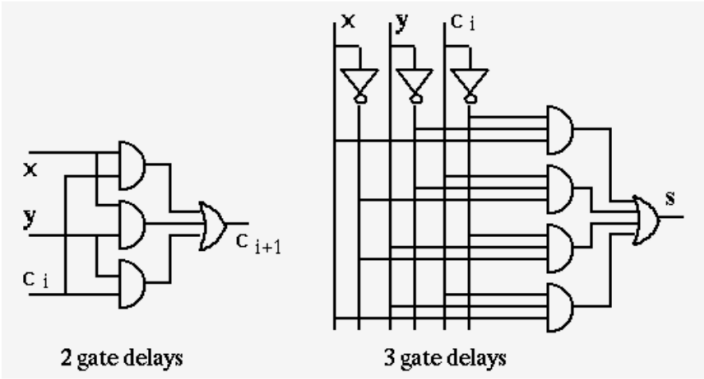
现在只需要来思考怎么处理进位的问题，行波进位是最直接的想法，每一位的进位是很好处理的，我们可以将低位进位输出传递给上一位作为其进位输入，也就会有:

$$s_i = a_i \oplus b_i \oplus c_{i-1}$$

$$c_i = a_i b_i + a_i c_{i-1} + b_i c_{i-1}$$

这就是所谓的**行波**，进位一级一级传递，就像波浪一样．但这是存在很明显的缺点的，由于下一位需要的进位输入需要上一位运算完成才能得到，因此会有非常夸张的门延迟:

行波进位加法器的缺点



$$\begin{cases} c_{i+1} = x_i y_i + x_i c_i + y_i c_i & 2\ gate\ delays \\ s_i = x_i' y_i' c_i + x_i' y_i c_i' + x_i y_i' c_i' + x_i y_i c_i & 3\ gate\ delays \end{cases}$$

对于一个 n 位行波进位加法器，其门延迟在最后一位达到最大，其进位输入需要等待前面所有门完成操作，$GateDelay = 2(n-1)+3 = (2n+1)gates$，这显然是不好的.

超前进位加法器-对于行波进位的改进

- 怎么改进？
- 消除延时
- 怎么消除延时？
- 考虑一下进位是哪来的，能不能展开这个级联的进位逻辑直接用所有位的输入得到任意位的进位输出？
- 当当！超前进位加法器!

我们考虑进位的产生:

- 当前位输入全 1 直接产生了进位，称为进位产生 (Carry Generated)，记为 $G_i = a_i b_i$
- 当前进位中至少有一个 1，从而只要上一位的运算产生进位，则进位会传播下去，称为进位传播 (Carry propagated)，记为 $P_i = a_i + b_i \ or \ P_i = a_i \oplus b_i$

那么以四位为例，其最后一位的进位输出就包括这些情况：最后一位发出了 G 信号；最后一位发出 P 信号，上一位产生进位；最后两位发出 P 信号，第二位产生进位...

$$c_{out} = G_i + P_i c_{in}$$

$$c_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 c_o$$

思考题

超前进位加法器完美吗？

hint

为什么文档中要把超前进位加法器再通过行波的方式进行串联？
Refence