中国科学技术大学
University of Science and Technology of China

数字电路实验

Lab6
超前进位加法器与ALU

2024/11/7

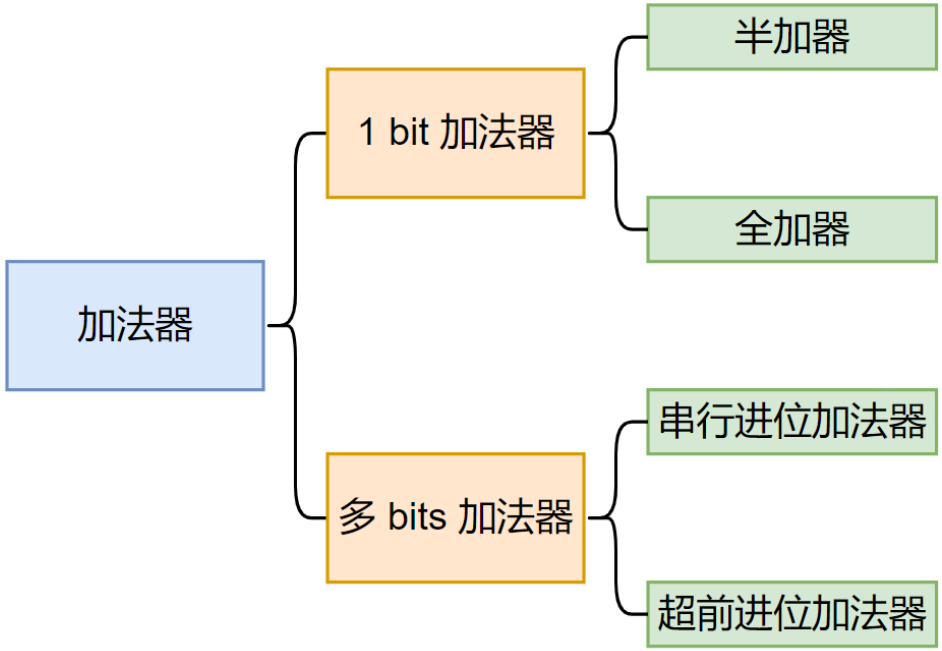
实验介绍

- 在本次实验中，我们将学习如何设计一个超前进位加法器，并最终使用Verilog实现。

- 我们还将介绍如何利用加法器设计一个ALU，并完成部分算数和逻辑运算。

加法器的实现方式

- **串行进位加法器**
  - 并行相加、串行进位，逻辑电路较简单
  - 高位的计算依赖低位的进位结果，电路延迟长、运算速度受限
- **超前进位加法器**
  - 在每一位上都进行预先进位，然后再进行加法运算
  - 每位的进位都只由加数和被加数唯一决定，与低位的进位无关



超前进位加法器

■ 设计流程

■ A+B的逻辑表达式：(S：结果，C：进位)

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

$$C_i = A_i B_i + (A_i \oplus B_i) C_{i-1}$$

■ 中间代换：(G，P)

$$G_i = A_i B_i,\ \ P_i = A_i \oplus B_i$$ ➡ 与A,B直接相关

■ 代入逻辑表达式：

$$S_i = P_i \oplus C_{i-1},\ \ C_i = G_i + P_i C_{i-1}$$ ➡ 仍与C_{i-1}直接相关

超前进位加法器

■ 设计流程

■ C_i逻辑表达式展开

$$C_0 = G_0 + P_0 C_{-1}$$

$$C_1 = G_1 + P_1 C_0 = G_1 + P_1 G_0 + P_1 P_0 C_{-1}$$

$$C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1}$$

$$C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{-1}$$

与或式，C_i只与G,P,C_{-1}直接相关

■ 将C_i结果带入以下表达式，得到S_i

$$S_i = P_i \oplus C_{i-1}$$

超前进位加法器

■ 设计流程

■ 根据C_i 、S_i逻辑表达式，编写Verilog代码

```
module Adder_LookAhead4 (
    input               [ 3 : 0]    a, b,
    input               [ 0 : 0]    ci,
    output              [ 3 : 0]    s,
    output              [ 0 : 0]    co
);

    wire    [3:0] C;
    wire    [3:0] G;
    wire    [3:0] P;

    assign  G = a & b;
    assign  P = a ^ b;

    assign  C[0] = G[0] | ( P[0] & ci );
    assign  C[1] = G[1] | ( P[1] & G[0] ) | ( P[1] & P[0] & ci );
    assign  C[2] = G[2] | ( P[2] & G[1] ) | ( P[2] & P[1] & G[0] ) | ( P[2] & P[1] & P[0] & ci );
    assign  C[3] = 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] & ci );

    assign  s[0] = P[0] ^ ci;
    assign  s[1] = P[1] ^ C[0];
    assign  s[2] = P[2] ^ C[1];
    assign  s[3] = P[3] ^ C[2];
    assign  co   = C[3];

endmodule
```

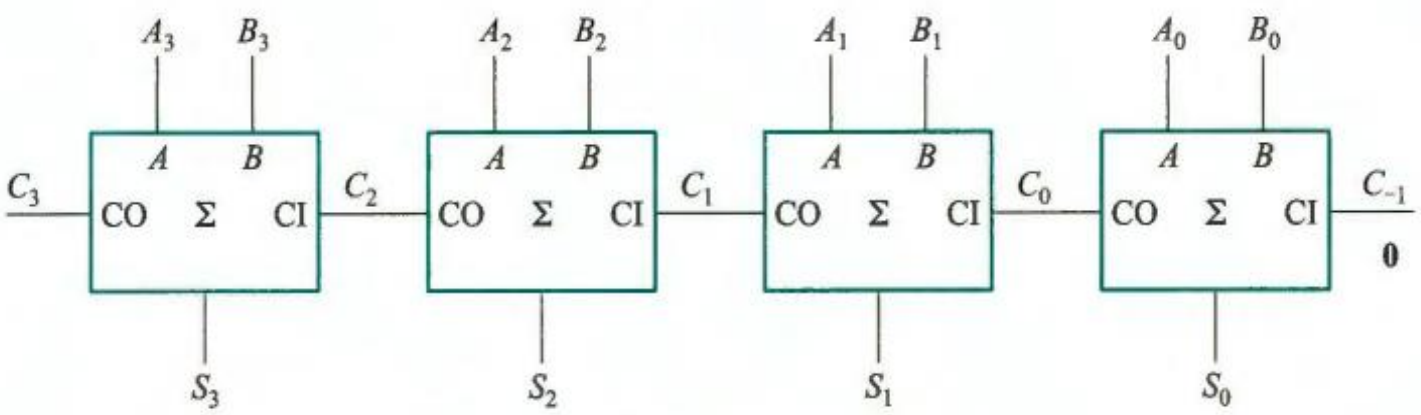
超前进位加法器

■ 结构优势

■ 串行进位加法器（4位）

- ✓ 每个加法器的进位信号C需要经过异或门、与门、或门各一个
- ✓ 若一个逻辑门的延迟为t_{pd}，那么一个加法器将产生$3t_{pd}$的延迟
- ✓ 四个加法器串行相连，共产生$12t_{pd}$的延迟

$$C_i = A_iB_i + (A_i \oplus B_i)C_{i-1}$$



超前进位加法器

■ 结构优势

■ 超前进位加法器（4位）

✓ 计算P,G经过了一级门电路

$$G_i = A_i B_i,\ P_i = A_i \oplus B_i$$

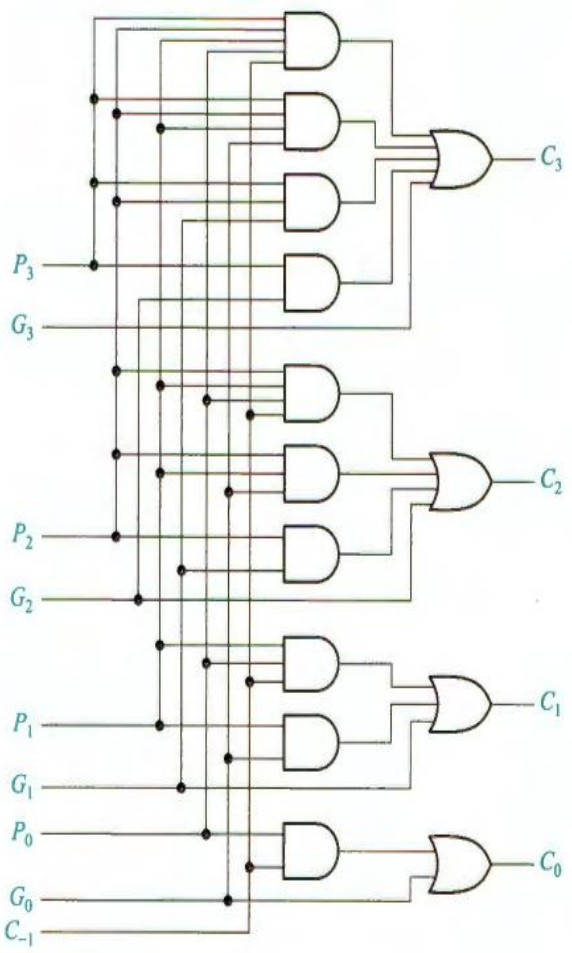
✓ 计算C共经过了与、或两级门电路

$$C_1 = G_1 + P_1 C_0 = G_1 + P_1 G_0 + P_1 P_0 C_{-1}$$

✓ 最后利用C与P计算出S，又经过了一级异或门电路

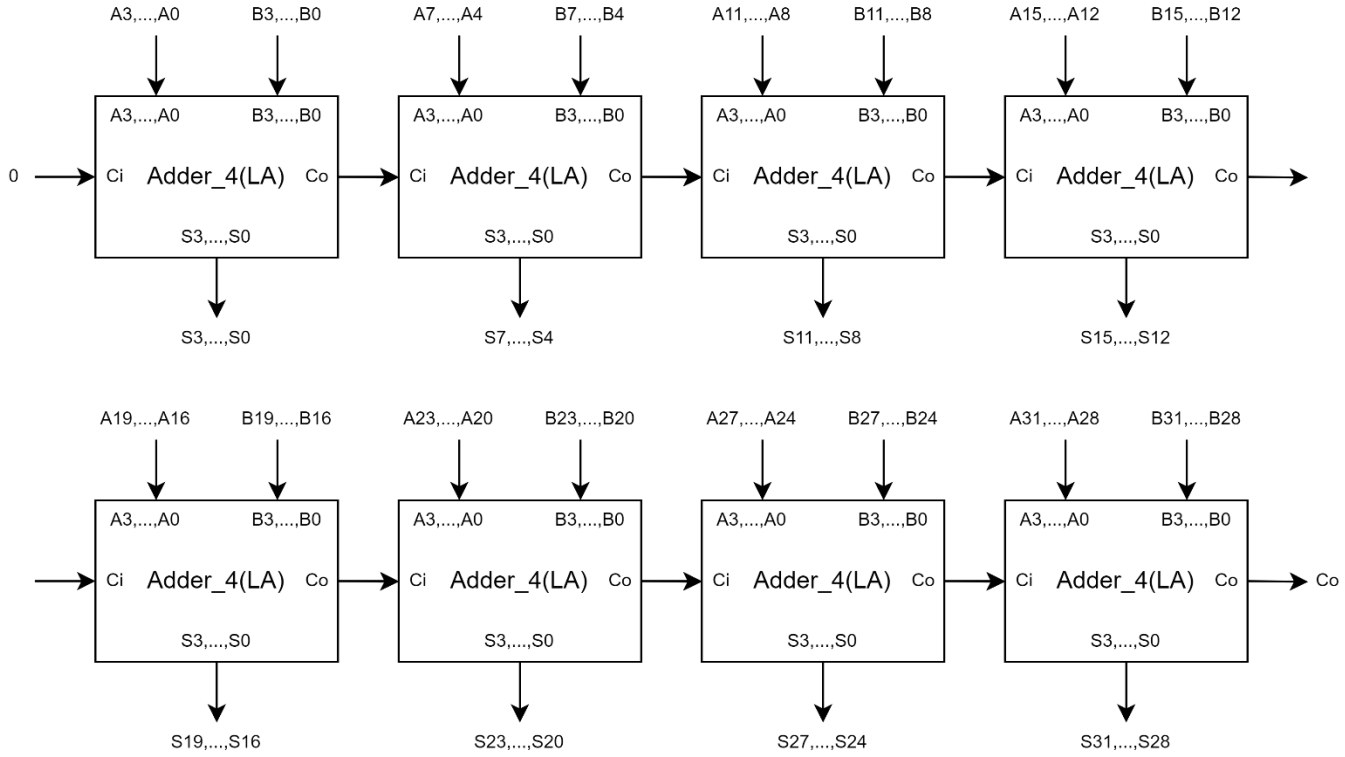
$$S_i = P_i \oplus C_{i-1}$$

✓ 共需要$4t_{pd}$的延迟



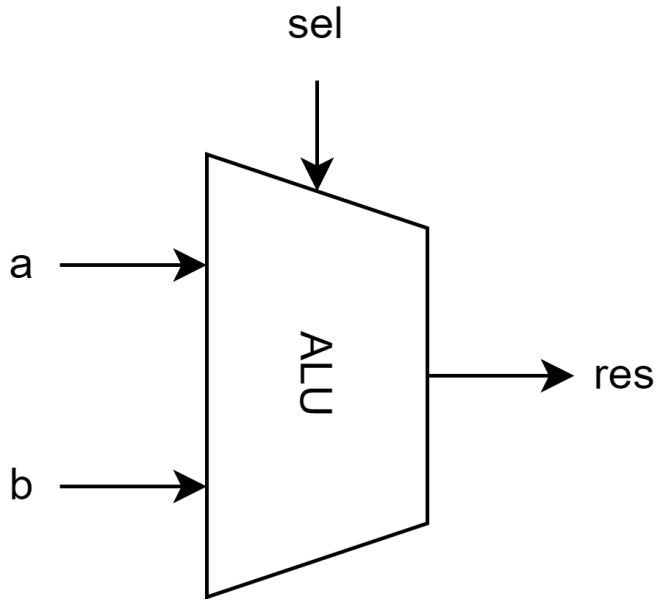
超前进位加法器

- 层次扩展
  - 借鉴串行进位加法器的思路，将多个超前进位加法器首尾相接串联起来
  - 以适量资源的增多，换取部分延迟的降低



ALU

- ALU(算术逻辑单元)
  - CPU 的一个重要部件，执行基本的算术、逻辑运算
  - 组合逻辑电路，要求ALU在一个周期内就能得到计算结果
  - 接受两个操作数a,b，以及一个独热码选择信号sel用以选择运算类型，输出为运算结果res



ALU

■ 龙芯架构 32 位精简版指令集中需要支持以下运算

■ 简单算术运算：

加法（ADD）、减法（SUB）、小于比较（SLT）
无符号小于比较（SLTU）

■ 逻辑运算：

按位或非（NOR）、按位与（AND）、按位或(OR)
按位异或（XOR），左移（SLL）、右移（SRL）
算术右移（SRA）

sel	res
12'h001	$a+b$
12'h002	$a-b$
12'h004	$a <_s b$
12'h008	$a <_u b$
12'h010	$a\&b$
12'h020	$a\|b$
12'h040	$\sim(a\|b)$
12'h080	$a\hat{}b$
12'h100	$a << b[4:0]$
12'h200	$a >> b[4:0]$
12'h400	$a >>> b[4:0]$
12'h800	b

ALU

■ 运算选择

- ■ 将两个源操作数送到所有运算单元
- ■ 通过输入信号sel，选择哪个运算单元的输出作为最终的结果

```
module ALU(
    input [31:0] a, b,
    input [11:0] sel,
    output [31:0] res
);
    //加法运算示例
    wire [31:0] adder_out;
    wire co;

    Adder #(WIDTH-1) fa1(
        .a(a),
        .b(b),
        .ci(0),
        .s(adder_out),
        .co(co)
    );

    //选择运算结果示例：加法
    assign res = ({32{sel[0]}} & adder_out)|...;

endmodule
```

■ 减法运算

■ 补码表示：负数由其相反数取反加一得到

$$\bar{x} = 2^n - 1 - x$$

■ 补码运算下的减法：

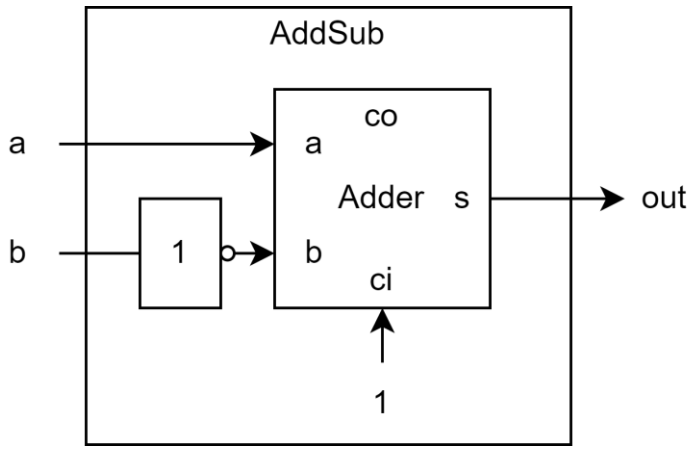
$$x' \equiv 2^n - x \equiv -x (\mathrm{mod}\ 2^n)$$

$$a - b = a + b'$$

■ 减法运算

■ 用加法器实现减法器

$$\text{out} = a + \bar{b} + 1 = a - b$$



AddSub.v

```
1  module AddSub (
2      input                [ 3 : 0]        a, b,
3      output               [ 3 : 0]        out,
4      output               [ 0 : 0]        co
5  );
6
7  Adder #(WIDTH-1) fa1(
8      .a(a),
9      .b(~b),
10     .ci(1'B1),
11     .s(out),
12     .co(co)
13 );
```

■ 溢出检测

- n 位补码表示的二进制数，其表示范围：

$$-2^{n-1} \sim 2^{n-1} - 1$$

- 超出了这个范围时，就发生了溢出

- 对于ALU中的加减法运算正确性无影响，但对需要用到加减运算结果的其他电路可能产生影响（如比较器）

ALU

■ 溢出检测

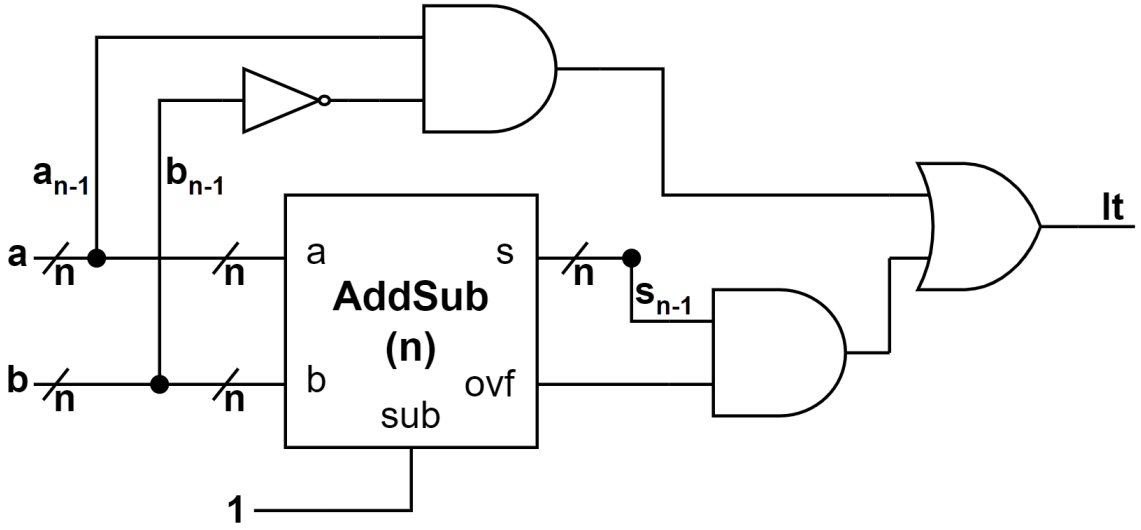
- ■ 加法：只有当a、b同号，结果异号时溢出
- ■ 减法：只有当a、b异号，结果与b同号时溢出

运算模式	a 的符号	b 的符号	异或后 b 的符号	结果的符号	是否溢出
加法 a+b	正	正	正	正	否
	正	**正**	**正**	**负**	**是**
	正	负	负	正	否
	正	负	负	负	否
	负	正	正	正	否
	负	正	正	负	否
	负	**负**	**负**	**正**	**是**
	负	负	负	负	否
减法 a-b	正	正	负	正	否
	正	正	负	负	否
	正	负	正	正	否
	正	**负**	**正**	**负**	**是**
	负	**正**	**负**	**正**	**是**
	负	正	负	负	否
	负	负	正	正	否
	负	负	正	负	否

ALU

■ 比较运算

- ■ 利用减法器设计比较器
- ■ a,b同号时：
  - ✓ 不发生溢出，a-b符号位为 1则代表a<b
- ■ a,b异号时：
  - ✓ 有可能发生溢出，直接比较符号位，正的大于负的



实验任务

- 【必做】学习超前进位加法器的设计方法。
- 【必做】在加法器的基础上，设计一个包括减法器、比较器在内的ALU。
- 【必做】完成 Lab6 实验练习题。

实验检查 DDL：11.14

报告提交 DDL：11.21

谢谢！