数据的表示和运算

马士兵教育研究院

目录

1.数制与编码

『进位计数制』及其相互转换

- 进位计数法

二进制（Binary）：逢2进一

0,1（最高位是符号位：0正，1负）

八进制（Octal）：逢8进一

0,1,2,3,4,5,6,7

十进制（Decimal）：逢10进一

0,1,2,3,4,5,6,7,8,9

十六进制（Hex）：逢16进一

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

任意进制转十进制：数码与权值相乘，再把乘积相加

二进制：	前缀：0b或0B	后缀：b或B
8位数	0b0000 0010 $(1*2^1+0*2^0=\mathbf{2})$ 0B1000 0010 $(1*2^1+0*2^0=\mathbf{-2})$ 0B0001 1011.1 = 27.5 $(1*2^4+1*2^3+0+1*2^1+1*2^0+1*2^{-1})$	
32位数	1000 0000 0000 0000 0000 0000 0000 0010b	
64位数	1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010B	

八进制：	前缀：0	后缀：o/O或q/Q
八进制举例：	012 $(1*8^1+2*8^0=\mathbf{10})$ 17O $(1*8^1+7*8^0=\mathbf{15})$	

十六进制：	前缀：0x或0X	后缀：h或H
十六进制举例：	0x00，0xFF（**255**）， 520H $(5*16^2+2*16^1+0=\mathbf{1312})$	

1.数制与编码

『进位计数制』及其相互转换

◆ 进制转换

二进制转八/十六进制

小数点前：右起往左数，三位/四位一组

小数点后：左起往右数，三位/四位一组

二转八:	点前左数，三位一组	点后右数，三位一组
01001110	001 001 110 1 1 6 => 116O	
11101.1011	011 101.101 100 3 5 . 5 4 => 35.54O	

二转十六:	点前左数，四位一组	点后右数，四位一组
01001110	0100 1110 4 E => 4EH	
11101.1011	0001 1101.1011 1 D . B => 1DBH	

1.数制与编码

『进位计数制』及其相互转换

◆ 进制转换

二进制转八/十六进制

十进制转其它进制

整数部分：**除基取余法**

小数部分：**乘基取整法**

将521.6875转二进制：

10 0000 1001.1011B

有些十进制小数无法完全转换为二进制

十转二：	整数：除基取余	小数：乘基取整法
521.6875	10 0000 1001	1011

除基		取余	
2	521	1	低位
2	260	0	
2	130	0	
2	65	1	
2	32	0	
2	16	0	
2	8	0	
2	4	0	
2	2	0	
2	1	1	高位
	0		

乘基	取整	
0.6875 x 2		
1.3750	1	高位
0.375 x 2		
0.750	0	
x 2		
1.50	1	
0.5 x 2		
1.0	1	低位

马士兵教育
www.mashibing.com

1.数制与编码

『进位计数制』及其相互转换

◆ 进制转换

二进制转八/十六进制

十进制转其它进制

整数部分：**除基取余法**

小数部分：**乘基取整法**

将521.6875转十六进制:

209.BH

有些十进制小数无法完全转换为十六进制

十转十六:	整数：除基取余	小数：乘基取整法
521.6875	209	B

	除基	取余	
16	521	9	低位
16	32	0	↓
16	2	2	高位
	0		

	乘基	取整	
	0.6875		高位
x	16		↓
	11.0	11	低位
	0		

1.数制与编码

『进位计数制』及其相互转换

◆ 进制转换

二进制转八/十六进制

十进制转其它进制

整数部分：**除基取余法**

小数部分：**乘基取整法**

将521.6875转八进制：

1011.54O

有些十进制小数无法完全转换为八进制

十转八：	整数：除基取余	小数：乘基取整法
521.6875	*1011*	*54*

除基		取余	
8	521	1	低位
8	65	1	↓
8	8	0	
8	1	1	高位
	0		

乘基	取整	
0.6875		
x 8		
5.5	5	高位
0.5		↓
x 8		
4.0	4	低位
0		

1.数制与编码

真值与『机器数』

◆ 真值

符号（+-） + 绝对值

◆ 机器数

符号和数值一起编码

无符号数:

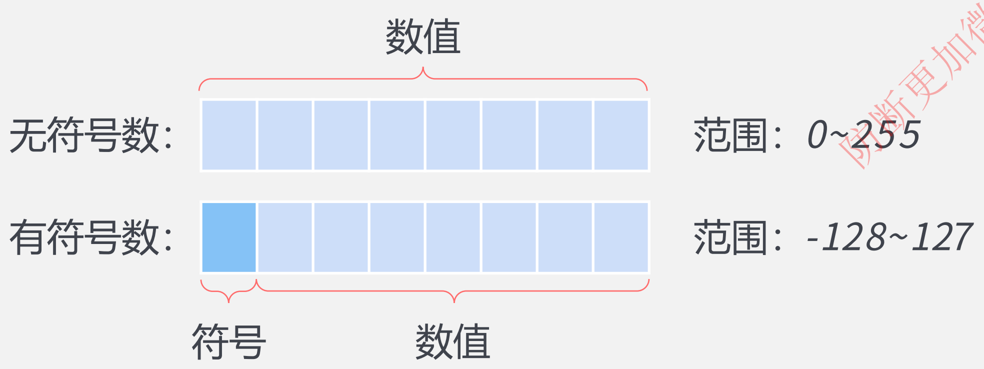
每个bit位都是数值位

只有正数和0

有符号数:

最高位代表符号：0正，1负

正负数各占一半（含0）



真值（无符号）	机器数（8位）	真值（有符号）
0	*0000 0000*	*+0*
127	*0111 1111*	*+127*
128	*1000 0000*	*-0*
255	*1111 1111*	*-127*

原码表示

1.数制与编码

真值与『机器数』

◆ 真值与机器数

真值：符号（+-） + 绝对值

机器数：符号和数值一起编码

原码的加法运算：

同号加，异号减；

取大绝对值符号

原码的减法运算：

同号减，异号加；

大绝对值 - 小绝对值；

取大绝对值符号

加法运算：

```
  0111 1111
+ 0000 0001
-----------
  1000 0000
```

```
  1111 1111
+ 0110 0001
-----------
          ?
```

```
  1111 1111
- 0110 0001
-----------
  1001 1110
```

减法运算：

```
  1111 1111
- 1000 0001
-----------
  1111 1110
```

```
  1111 1111
- 0000 0001
-----------
          ?
```

```
  1111 1111
+ 0000 0001
-----------
 10000 0000
  1000 0000
```

-128?
Yes!

真值（无符号）	机器数（8位）	真值（有符号）
0	*0000 0000*	*+0*
127	*0111 1111*	*+127*
128	*1000 0000*	*-128*
255	*1111 1111*	*-127*

1.数制与编码

真值与『机器数』

- 机器数

  正数的原码、反码、补码都一样

  原码：符号位 + 数值位

  -反码：符号位不变，数值位取反

  补码：反码 + 1

真值	原码	反码	补码
127	*0111 1111*	*0111 1111*	*0111 1111*
-1	*1000 0001*	*1111 1110*	*1111 1111*
-81	*1101 0001*	*1010 1110*	*1010 1111*
-127	*1111 1111*	*1000 0000*	*1000 0001*

真值（无符号）	机器数（8位）	真值（有符号）
0	*0000 0000*	*+0*
127	*0111 1111*	*+127*
128	*1000 0000*	*-128*
255	*1111 1111*	*-1*
129	*1000 0001*	*-127*

补码表示

1.数制与编码

真值与『机器数』

◆ 机器数

正数的原码、反码、补码都一样

原码：符号位 + 数值位

-反码：符号位不变，数值位取反

补码：反码 + 1

计算机中数的存储和运算，都使用补码

加法运算：
A+B

```
  0111 1111
+ 0000 0001
-----------
  1000 0000
```

```
  1111 1111
+ 0110 0001
-----------
 10110 0000
  0110 0000
```

```
  1111 1011
+ 1111 1101
-----------
 11111 1000
  1111 1000
```

减法运算：
A-B=A+(-B)

```
  1111 1111
- 1000 0001
-----------
  0111 1110
```

```
  1111 1111
+ 0111 1111
-----------
 10111 1110
  0111 1110
```

```
  1000 0001
- 0000 0001
-----------
  1000 0000
```

真值（无符号）	机器数（8位）	真值（有符号）
0	*0000 0000*	*+0*
127	*0111 1111*	*+127*
128	*1000 0000*	*-128*
255	*1111 1111*	*-1*
129	*1000 0001*	*-127*

补码表示

1.数制与编码

字符与汉字编码

- **字符编码**

  ASCII：美国标准信息交换码

  共128个字符

- **汉字编码**

  GBK2312，GB18030，GBK

  一个汉字占两个字节

  UTF-8/UTF8：

  国际通用字符集，兼容ASCII码

  使用1-4个可变长度字节编码

ASCII 码表

ASCII 值	控制字符	ASCII 值	控制字符	ASCII 值	控制字符	ASCII 值	控制字符
0	NUT	32	(space)	64	@	96	、
1	SOH	33	!	65	A	97	a
2	STX	34	”	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	,	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	－	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DCI	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	X	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	/	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	ˆ	126	~
31	US	63	?	95	—	127	DEL

马士兵教育
www.mashibing.com

目录

2.数的表示和运算

『定点数』的表示

◆ 定点数

小数点位置是**固定**的，但不需要点号

定点小数：即纯小数

小数点在符号位之后，有效数值位之前

范围（补码）：$-1 \le x \le 1 - 2^{-n}$

定点整数：即纯整数

小数点在有效数值位之后

范围（补码）：$-2^{n} \le x \le 2^{n} - 1$

优点：定点运算简单，实现容易

缺点：表示范围小，运算精度低，不适合科学运算

定点小数

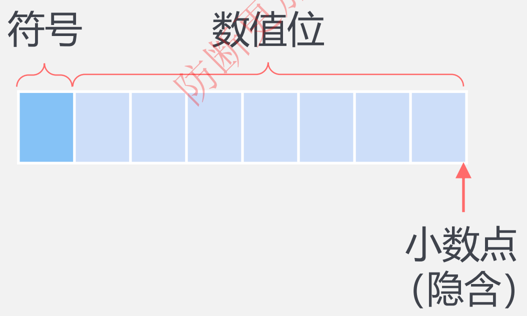
符	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
1	1	1	0	1	0	0	0

$[x]_{原} = 1.1101000$

$[x]_{补} = 1.0011000$

真值：*-0.8125*

定点整数



符	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
0	0	0	1	1	0	1	0

$[x]_{原} = 0,0011010$

$[x]_{补} = 0,0011010$

真值：*26*

2.数的表示和运算

『定点数』的运算

◆ 定点数

定点整数的运算：

算数运算：加减乘除（补码）

把减法转成加法：A-B=A+(-B)

取反加1

B → **-B**

B	-B
补码： 1000 0111	**补码：** 0111 1001
反码： 1000 0110	**反码：** 0111 1001
原码： 1111 1001	**原码：** 0111 1001

加法运算：
A+B

```
  0111 1111
+ 0000 0001
-----------
  1000 0000
```

```
   1111 1111
 + 0110 0001
------------
  10110 0000
   0110 0000
```

```
   1111 1011
 + 1111 1101
------------
  11111 1000
   1111 1000
```

减法运算：
A-B=A+(-B)

```
  1111 1111
- 1000 0001
-----------
  0111 1110
```

→

```
   1111 1111
 + 0111 1111
------------
  10111 1110
   0111 1110
```

```
  0111 1111
- 1000 0111
-----------
          ?
```

→

```
  0111 1111
+ 0111 1001
-----------
  1111 1000
```

2.数的表示和运算

『定点数』的运算

◆ 定点数

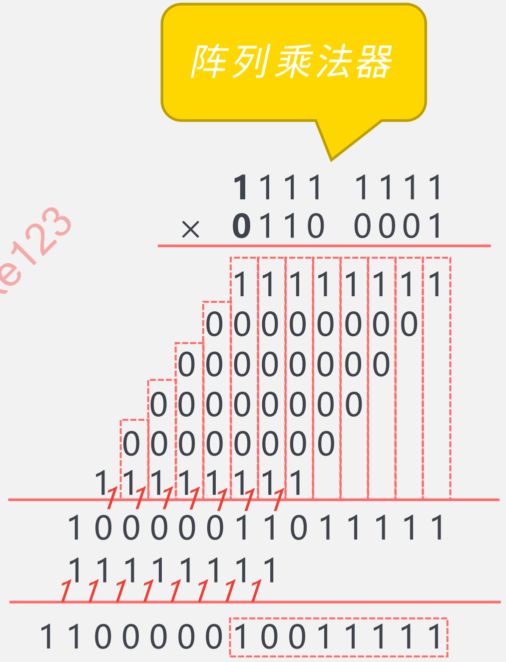
定点整数的运算：

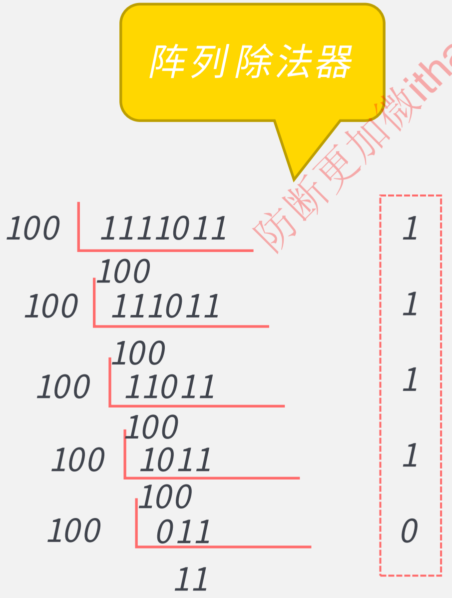
算数运算：加减乘除（补码）

把乘法转成累加

把除法转成累减

减->加





2.数的表示和运算

『定点数』的运算

◆ 定点数

定点整数的运算：

算数运算：+、-、×、/

逻辑运算：与&、或| 、非!

按位运算：&、|、^、~

按位与&：同1则1，不同则0

按位或|：有1则1

按位异或^：同0，异1

按位取反~：0变1，1变0

逻辑与&：
同1则1，不同则0

1 & 1 = 1
1 & 0 = 0
0 & 1 = 0

逻辑或|：
有1则1，同0则0

1 | 1 = 1
1 | 0 = 1
0 | 1 = 1
0 | 0 = 0

逻辑非!：
非1则0，非0则1

!1 = 0
!0 = 1

```
  1111 1111
& 0110 0001
-----------
  0110 0001
```

```
  1100 1111
& 0000 0001
-----------
  0000 0001
```

```
  1100 1110
& 0000 0001
-----------
  0000 0000
```

```
  1111 1111
| 0110 0001
-----------
  1111 1111
```

```
  1111 1111
^ 0110 0001
-----------
  1001 1110
```

```
  1001 1110
^ 0110 0001
-----------
  1111 1111
```

```
~ 0110 0001
-----------
  1001 1110
```

2.数的表示和运算

『定点数』的运算

◆ 定点数

定点整数的运算：

算数运算：+、-、×、/

逻辑运算：&、 | 、！

按位运算：&、|、^、~、<<、>>、>>>

左移<<：向左平移，右边补0

右移>>：向右平移，左边补符号位

无符号右移>>>：向右平移，左边补0

```
  1111 1111        1100 1111        1100 1110
& 0110 0001      & 0000 0001      & 0000 0001
-----------      -----------      -----------
  0110 0001        0000 0001        0000 0000

  1111 1111        1111 1111        1001 1110
| 0110 0001      ^ 0110 0001      ^ 0110 0001
-----------      -----------      -----------
  1111 1111        1001 1110        1111 1111

~ 0110 0001
-----------
  1001 1110
```

0110 0001<<3位 ⟶ 0110 0001 $\times 2^{11}$ ⟶ 011 $\boxed{0000\ 1000}$

0110 0001>>3位 ⟶ 0110 0001 $\times 2^{-101}$ ⟶ $\boxed{0000\ 1100}$ 001

1110 0001>>3位 ⟶ 1110 0001 $\times 2^{-101}$ ⟶ $\boxed{1111\ 1100}$ 001

1110 0001>>>3位 ⟶ 1110 0001 $\times 2^{-101}$ ⟶ $\boxed{0001\ 1100}$ 001

2.数的表示和运算

『浮点数』的表示

- 浮点数

  小数点位置是**不固定**的，根据需要浮动

  任何一个R进制数N，可以表示为：

  $$(N)_R = \pm S \times R^{\pm e}$$

  S-尾数：N的有效数字，反映了数的精度

  常用补码表示的**定点小数**

  尽可能占满尾数，保留更多有效数字

  R-基值：即进制数，2/4/8/16等

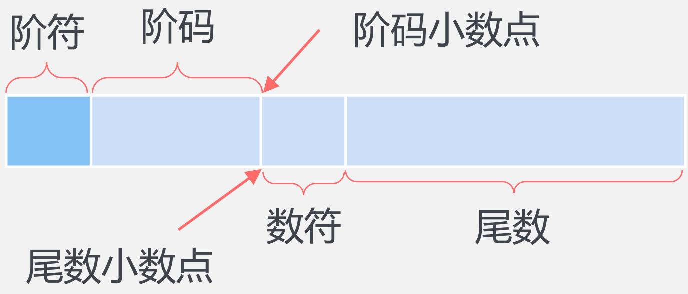
  e-阶码：小数点的实际位置，反映了表示范围

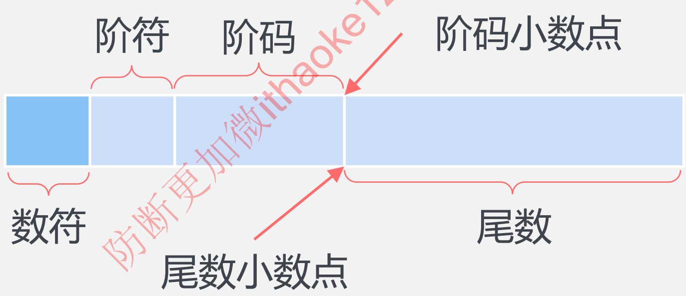
  常用补码表示的**定点整数**

  数符；阶符

$$(N)_{10} = 3.1415926 = 314.15926 \times 10^{-2} = 0.031415926 \times 10^{2}$$

$$(N)_2 = 1101.1001 = 11011001 \times 2^{-100} = 0.11011001 \times 2^{100}$$





举例：

a=0,01；1.1001

0	0	1	1	1	0	0	1

阶码：+1

尾数：-0.1001(补)=-0.0111(原)

真值：$-0.0111 \times 2^1 = -0.111$

b=0,10；0.01001

0	1	0	0	0	1	0	0	1

阶码：+2

尾数：$0.01001 = 0.01001 \times 2^2 = +1.001$

2.数的表示和运算

『浮点数』的表示

◆ 浮点数

规格化：尾数的最高数值位必须有效（非0）

左规：最高数值位无效，尾数左移，阶码减1

右规：尾数“假溢出”，右移，阶码加1

$(N)_{10} = 3.1415926 = 314.15926 \times 10^{-2} = 0.031415926 \times 10^{2}$

$(N)_{2} = 1101.1001 = 11011001 \times 2^{-100} = 0.11011001 \times 2^{100}$

b=0,10 ; 0.01001

尾数：0.01001$\times 2^2$=0.1001$\times 2^1$

阶码：+1

0	0	1	0	0	1	0	0

规格化：左规

c=+1.0100$\times 2^2$ =0.1010 $\times 2^3$

0	1	1	0	1	0	1	0

规格化：右规

举例：

a=0,01 ; 1.1001

0	0	1	1	1	0	0	1

阶码：+1

尾数：-0.1001(补)=-0.0111 (原)

真值：-0.0111 $\times 2^1$ =-0.111

b=0,10 ; 0.01001

0	1	0	0	0	1	0	0	1

阶码：+2

尾数：0.01001=0.01001$\times 2^2$=+1.001

2.数的表示和运算

『浮点数』的运算

◆ 步骤

1. 对阶：使阶码相等（小->大）
2. 尾数求和/差
3. 规格化
4. 舍入：

   截断法（恒舍法）：强制舍去

   0舍1入法：舍弃位的最高位为1，则进1

   恒置1法：末位恒置1
5. 判断溢出：

   阶码上溢（异常）；阶码下溢（作0）

举例：十进制数X= -7/256，Y= +59/1024，按机器补码浮点运算计算X-Y，结果用二进制表示。浮点数格式如下：2位阶符，3位阶码，2位数符，9位尾数。

答：用补码表示阶码和尾数：

X：E= 1/256=2^{-8}，S=-7=-111，N=-111 × 2^{-8}=-0.111 × 2^{-5}

=-0.111 × 2^{-101}=-0.001 × 2^{-011}（补）

=11011,11.001000000

1	1	0	1	1	1	1	0	0	1	0	0	0	0	0	0

Y：E=1/1024= 2^{-10}，S=+59=+111011，N=+111011 × 2^{-10}

=+0.111011 × 2^{-4}=+0.111011 × 2^{-100}（补）

=11100,00.111011000

1	1	1	0	0	0	0	1	1	1	0	1	1	0	0	0

-1

求阶差：$[E_X]$- $[E_Y]$=11011-11100=11011+00100=11111<0

对阶：$[E_X]$=11011 -11111=11100,X=11100,11.100100000

尾数加减：$[S_X]$-$[S_Y]$=$[S_X]$ +$[-S_Y]$=11.100100000+11.000101000

=10.101001000（10：尾数溢出，需要进行规格化）

规格化：右规（尾数右移，阶码+1）-> 11.010100100（高位补符号1）

X-Y = 11101, 11.010100100（-0.010100100 × 2^{-101} ）

1	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0

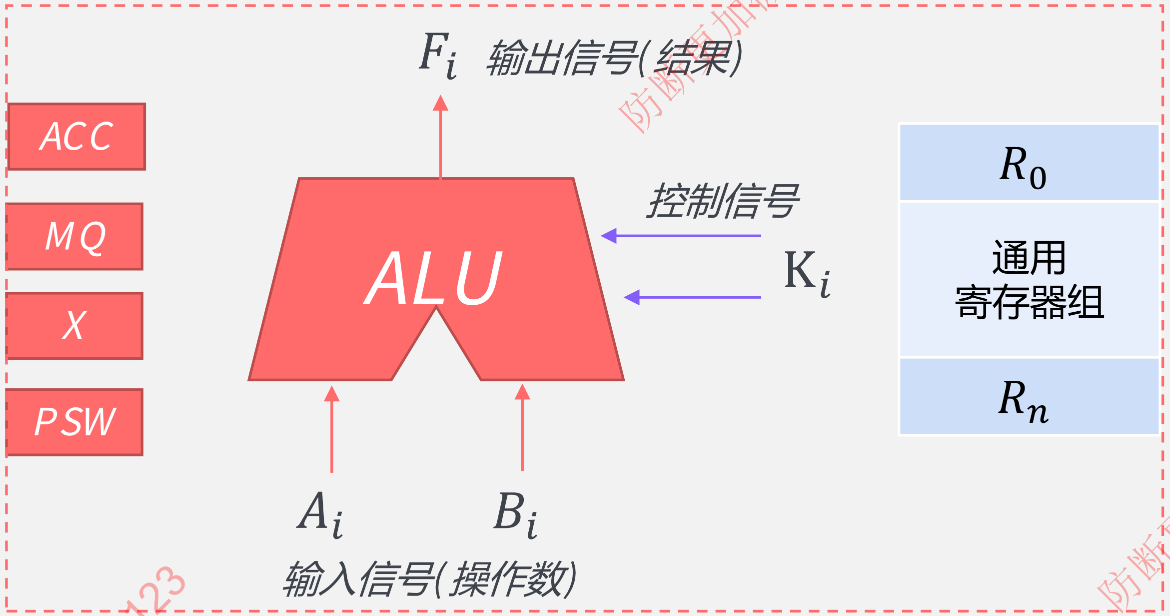
目录

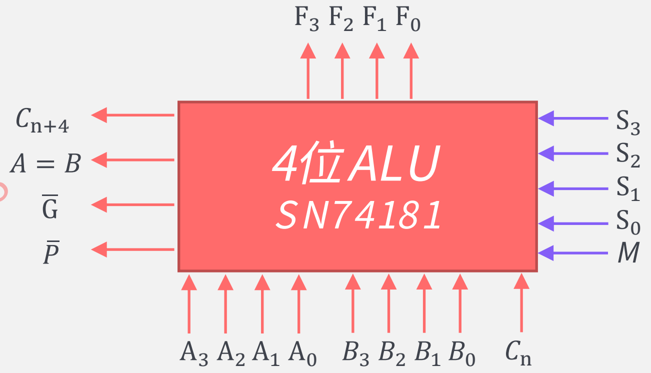
3.算数逻辑单元ALU

ALU的功能和结构

- 算数运算归结为加法运算
- 控制信号决定是否进行逻辑运算

算数运算：加/减/乘/除
逻辑运算：与/或/非
辅助功能：移位/求补





3.算数逻辑单元ALU

ALU的功能和结构

◆ 基本逻辑运算的实现

与&: $Y = A \cdot B$

或|: $Y = A + B$

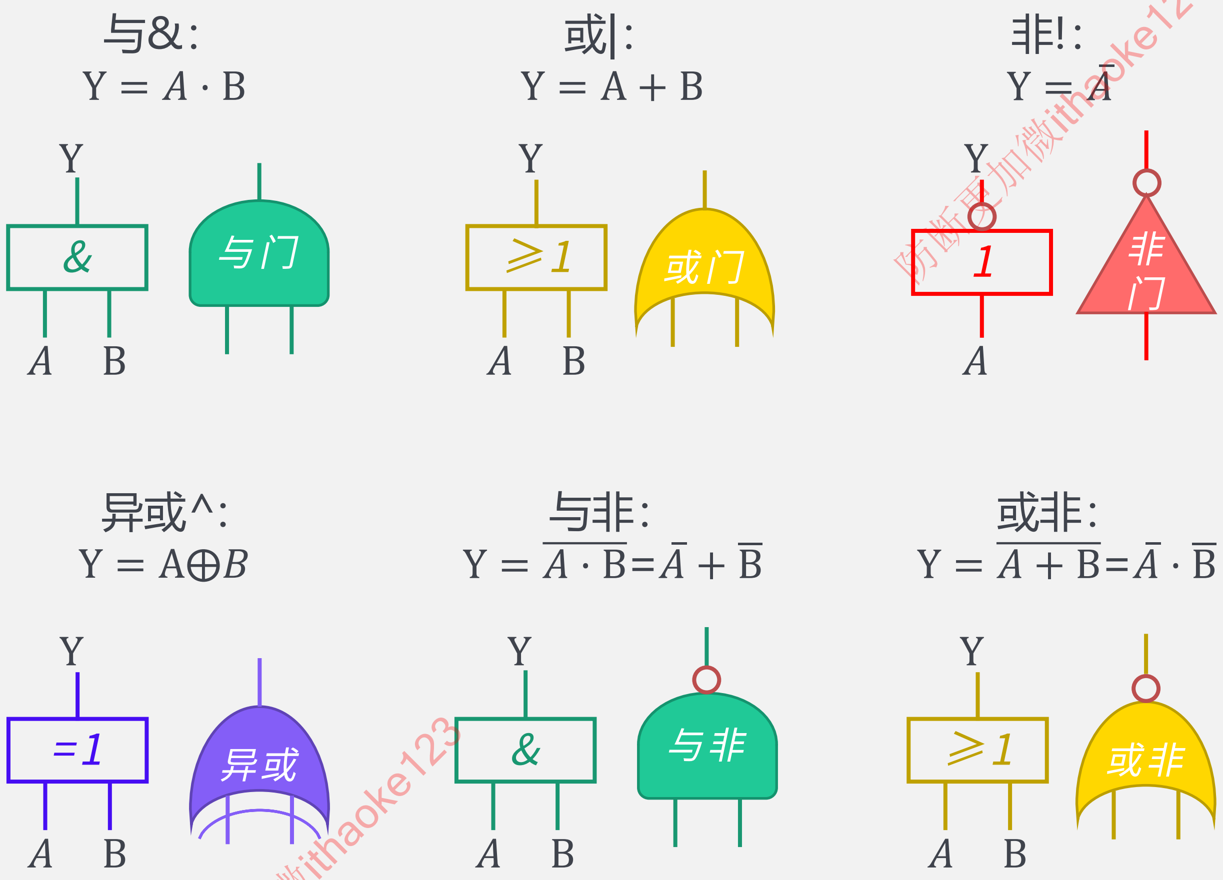
非!: $Y = \bar{A}$

异或^: $Y = A \oplus B$

与非: $Y = \overline{A \cdot B} = \bar{A} + \bar{B}$

或非: $Y = \overline{A + B} = \bar{A} \cdot \bar{B}$

	异或				与非				或非			
A	0	0	1	1	0	0	1	1	0	0	1	1
B	0	1	0	1	0	1	0	1	0	1	0	1
Y	0	1	1	0	1	1	1	0	1	0	0	0



3.算数逻辑单元ALU

串行/并行加法器

- **一位全加器**
- 串行加法器
- 并行加法器

$$
\begin{array}{r}
1111\ \ 1111 \\
+\ 0110\ \ 0001 \\
\hline
11111\ \ \ 111 \\
\boxed{1}0110\ \ 0000
\end{array}
$$

A_i：操作数，本位

B_i：操作数，本位

C_{i-1}：低位的进位

S_i：本位的和

	1	1	1	1	1	1	1	1	A_i	输入
	0	1	1	0	0	0	0	1	B_i	
	1	1	1	1	1	1	1	0	C_{i-1}	
	0	1	1	0	0	0	0	0	S_i	输出
1	1	1	1	1	1	1	1	0	C_i	

1.怎么计算 S_i？

S_i：输入中有奇数个1时则为1，否则为0

异或^：1^0^0=1,0^1^1=0

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

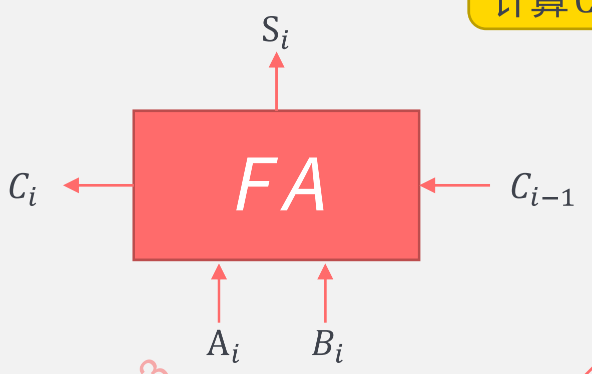
2.怎么计算 C_i？

C_i：输入中至少有两个1则为1，否则为0

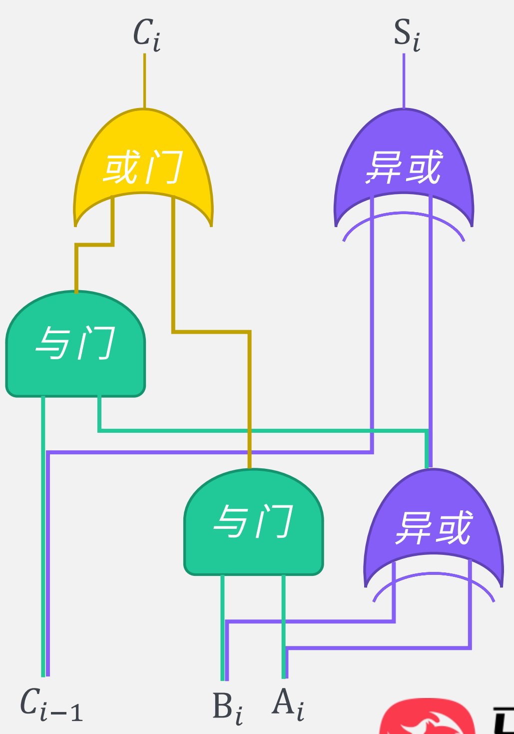
与&：A_i 和 B_i 同时为1，

或者：A_i 和 B_i 有一个为1，同时 C_{i-1} 为1

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



一位全加器：*FA，Full Adder*



3.算数逻辑单元ALU

串行/并行加法器

- 一位全加器
- 串行加法器

  进位触发器：保存 进位 位

  数据逐位送入FA中运算

  串行、逐位送回寄存器
- 并行加法器

$$
\begin{array}{rl}
 & 1111\ 1111 \\
+ & 0110\ 0001 \\
 & \scriptstyle 1\,1\,1\,1\,1\ \ 1\,1\,1 \\
\hline
 & \boxed{1}0110\ 0000
\end{array}
$$

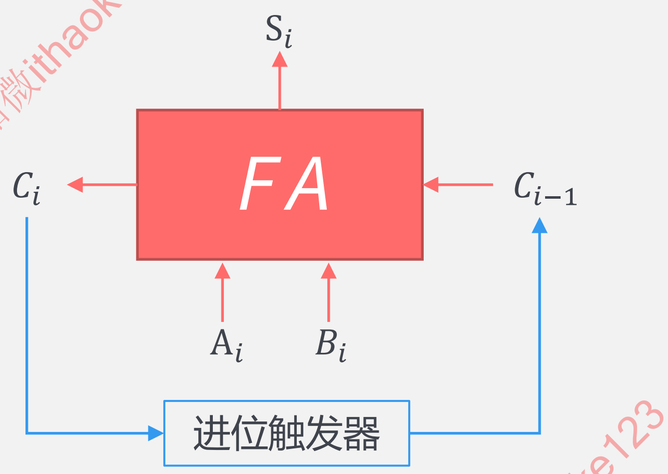
A_i：操作数，本位

B_i：操作数，本位

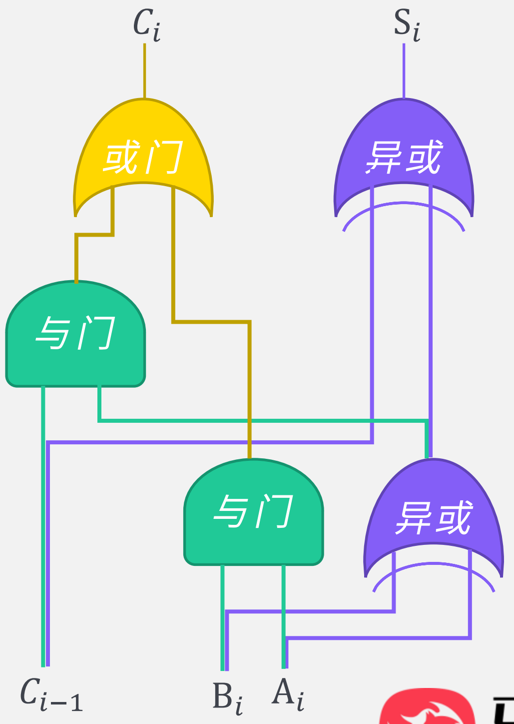
C_{i-1}：低位的进位

S_i：本位的和

	1	1	1	1	1	1	1	1	A_i	输入
	0	1	1	0	0	0	0	1	B_i	
	1	1	1	1	1	1	1	0	C_{i-1}	
	0	1	1	0	0	0	0	0	S_i	输出
1	1	1	1	1	1	1	1	0	C_i	



每次产生一位和，共需操作n次



3.算数逻辑单元ALU

串行/并行加法器

- 一位全加器
- 串行加法器
- **并行加法器**

把n个全加器串接起来

串行进位：进位信号逐级形成

运算速度取决于每一个进位的产生速度

```
  1111 1111     Ai：操作数，本位
+ 0110 0001     Bi：操作数，本位
  11111  111    Ci-1：低位的进位
 10110 0000     Si：本位的和
```

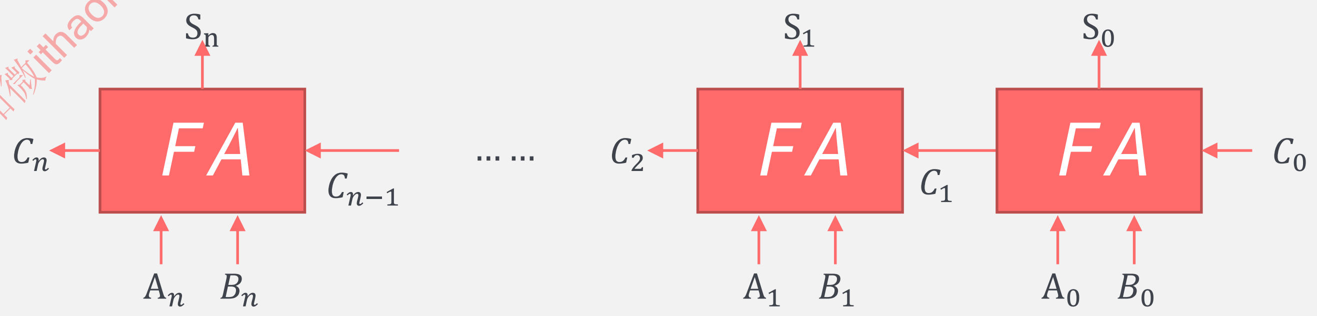
A_i：操作数，本位

B_i：操作数，本位

C_{i-1}：低位的进位

S_i：本位的和

	1	1	1	1	1	1	1	1	A_i	输入
	0	1	1	0	0	0	0	1	B_i	
	1	1	1	1	1	1	1	0	C_{i-1}	
	0	1	1	0	0	0	0	0	S_i	输出
1	1	1	1	1	1	1	1	0	C_i	



3.算数逻辑单元ALU

串行/并行加法器

- 一位全加器
- 串行加法器
- **并行加法器**

把n个全加器串接起来

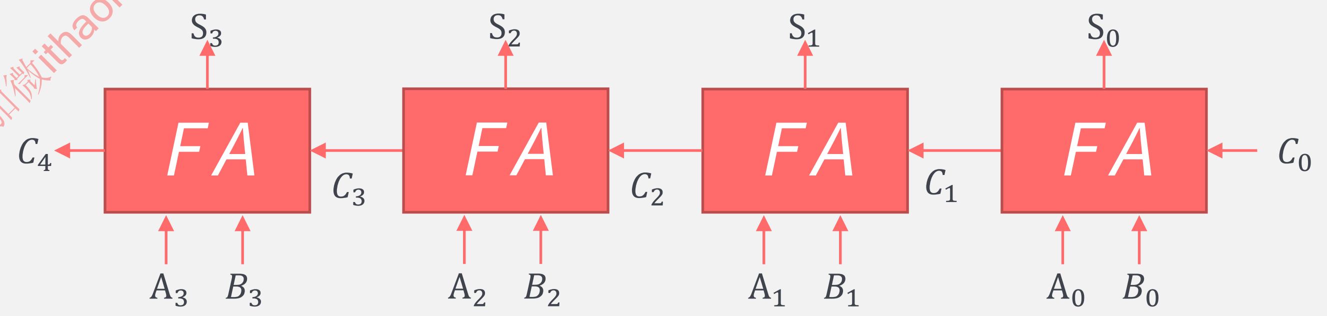
串行进位：进位信号逐级形成

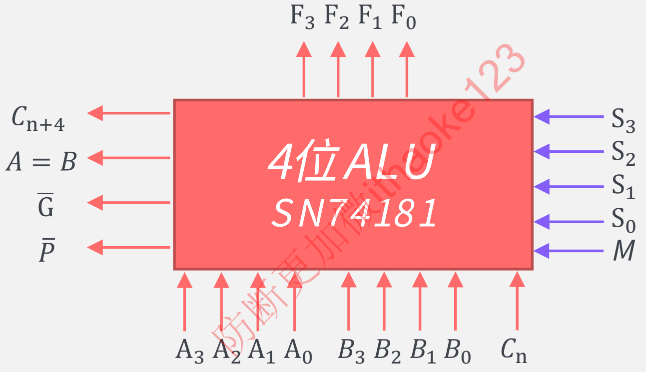
运算速度取决于每一个进位的产生速度

改进：A_n~A_0，B_n~B_0是已知的

```
   1111 1111     A_i：操作数，本位
 + 0110 0001     B_i：操作数，本位
  1 1 1 1 1  1 1 1   C_{i-1}：低位的进位
  10110 0000     S_i：本位的和
```

	1	1	1	1	1	1	1	1	A_i	输入
	0	1	1	0	0	0	0	1	B_i	
	1	1	1	1	1	1	1	0	C_{i-1}	
	0	1	1	0	0	0	0	0	S_i	输出
1	1	1	1	1	1	1	1	0	C_i	





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

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

$G_i = A_i B_i$

$P_i = A_i \oplus B_i$

3.算数逻辑单元ALU

串行/并行加法器

- 一位全加器
- 串行加法器
- **并行加法器**

  把n个全加器串接起来

  串行进位：进位信号逐级形成

  运算速度取决于每一个进位的产生速度

  改进：$A_n \sim A_0$，$B_n \sim B_0$是已知的

$$
\begin{array}{rl}
 & 1111\ \ 1111 \\
+ & 0110\ \ 0001 \\
 & {\scriptstyle 11111\ \ \ 111} \\
\hline
 & \boxed{1}0110\ \ 0000
\end{array}
$$

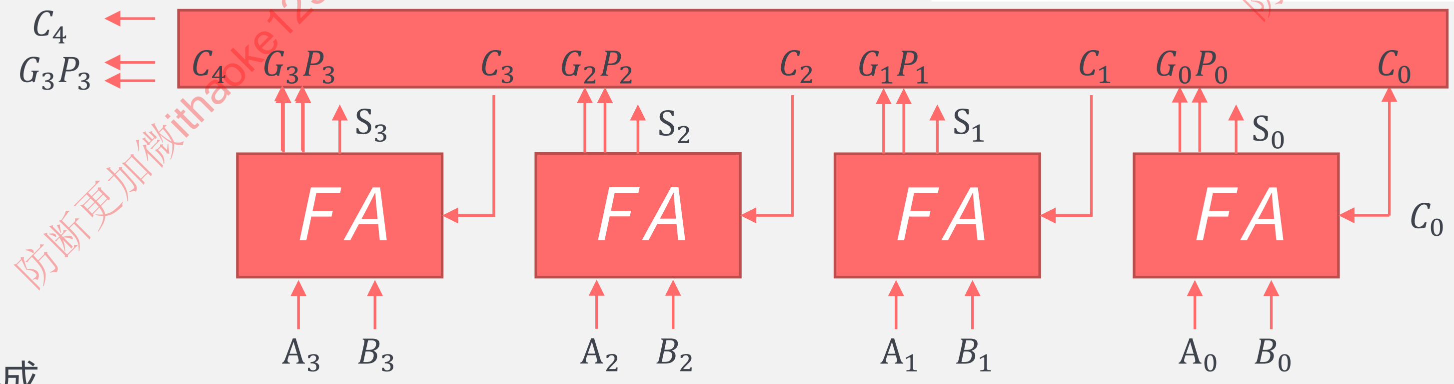
A_i：操作数，本位

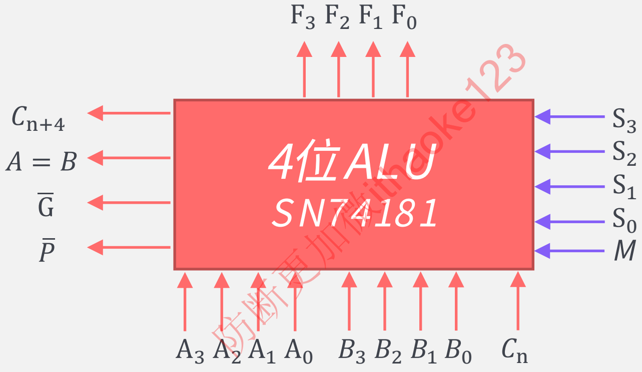
B_i：操作数，本位

C_{i-1}：低位的进位

S_i：本位的和

	1	1	1	1	1	1	1	1	A_i	输入
	0	1	1	0	0	0	0	1	B_i	
	1	1	1	1	1	1	1	0	C_{i-1}	
	0	1	1	0	0	0	0	0	S_i	输出
1	1	1	1	1	1	1	1	0	C_i	





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

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

$G_i = A_iB_i$

$P_i = A_i \oplus B_i$

3.算数逻辑单元ALU

串行/并行加法器

- 一位全加器
- 串行加法器
- **并行加法器**

把n个全加器串接起来

串行进位：进位信号逐级形成

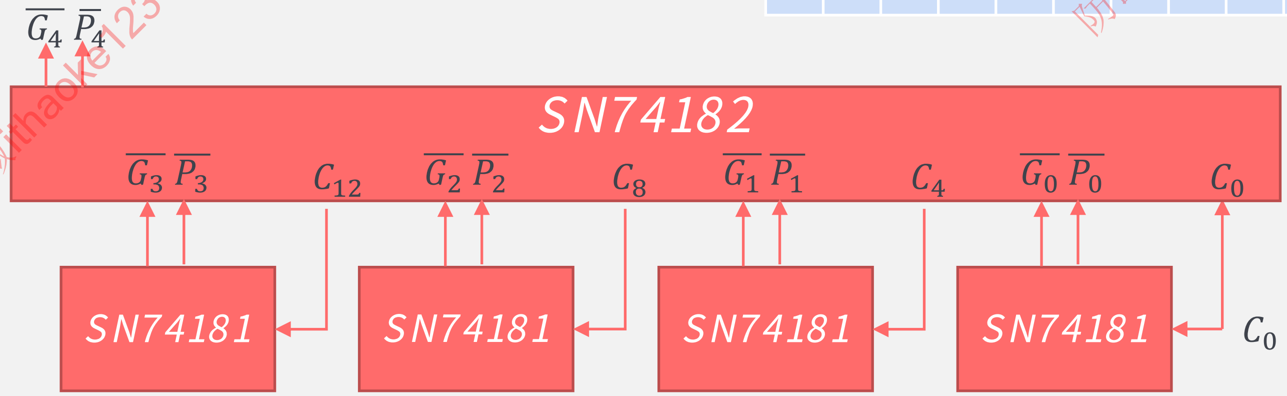
运算速度取决于每一个进位的产生速度

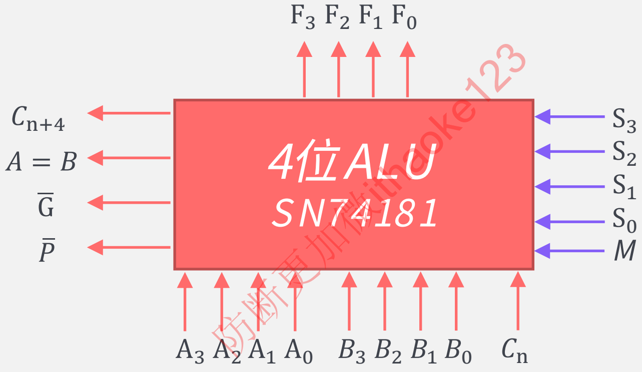
改进：$A_n \sim A_0$，$B_n \sim B_0$是已知的

组内并行、组间并行

$$
\begin{array}{rll}
 & 1111\ 1111 & A_i\text{：操作数，本位} \\
+ & 0110\ 0001 & B_i\text{：操作数，本位} \\
 & 11111\ \ 111 & C_{i-1}\text{：低位的进位} \\
\hline
 & \boxed{1}0110\ 0000 & S_i\text{：本位的和}
\end{array}
$$

	1	1	1	1	1	1	1	1	A_i	输入
	0	1	1	0	0	0	0	1	B_i	
	1	1	1	1	1	1	1	0	C_{i-1}	
	0	1	1	0	0	0	0	0	S_i	输出
1	1	1	1	1	1	1	1	0	C_i	





马士兵教育

www.mashibing.com

扫码加马老师微信