4.5 常用组合逻辑电路

4.5.4 数值比较器

定义：它对两个数进行比较，判断其大小的逻辑电路。此时，0、1将有大小、正负的区别。

（1） 1位数值比较器(设计)

输入：两个1位二进制数 A、B。

输出：$F_{A>B}=1$表示A大于B

$F_{A<B}=1$表示A小于B

$F_{A=B}=1$表示A等于B

4.5.4 数值比较器——1位数值比较器

功能表

输入		输出		
A	B	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

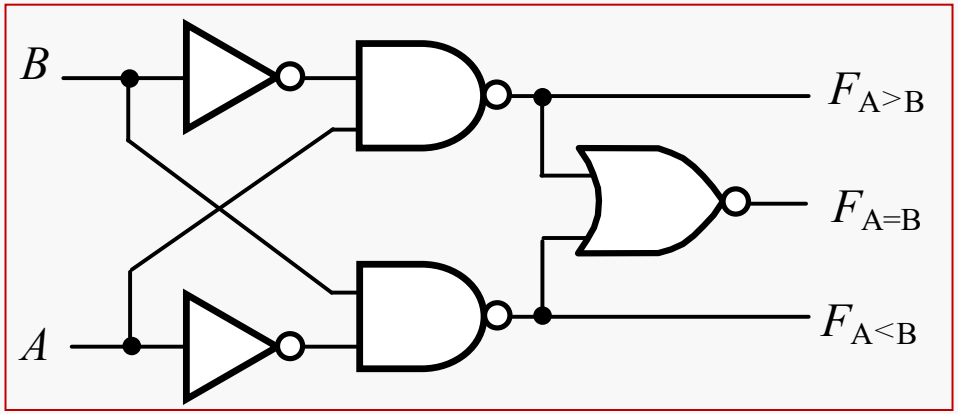
逻辑表达式

$$F_{A>B} = A\overline{B}$$

$$F_{A<B} = \overline{A}B$$

$$F_{A=B} = \overline{A}\,\overline{B} + AB$$

逻辑图



4.5.4 数值比较器——2位数值比较器

输入：两个2位二进制数 $A=A_1A_0$、$B=B_1B_0$。

输出：$F_{A>B}$、$F_{A<B}$、$F_{A=B}$

常规方法：

根据A、B大小关系，填写4输入真值表，然后进行设计。

数值大小的比较方法：

（1）先对高位进行比较，当（A_1、B_1）不相等时，高位比较的结果就是两个数的比较结果；

（2）当高位相等时，再比较低位（A_0、B_0）的大小；

（3）更多位的数值比较也**以此类推**。

扩展的方法：

在1位数值比较器的基础上，扩展实现2位数值比较器。

4.5.4 数值比较器——2位数值比较器

功能表

输　入				输　出		
A_1	B_1	A_0	B_0	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_1 > B_1$		×		1	0	0
$A_1 < B_1$		×		0	1	0
$A_1 = B_1$		$A_0 > B_0$		1	0	0
$A_1 = B_1$		$A_0 < B_0$		0	1	0
$A_1 = B_1$		$A_0 = B_0$		0	0	1

逻辑表达式

$$F_{A>B} = (A_1>B_1) + (A_1=B_1)(A_0>B_0)$$

$$F_{A<B} = (A_1<B_1) + (A_1=B_1)(A_0<B_0)$$

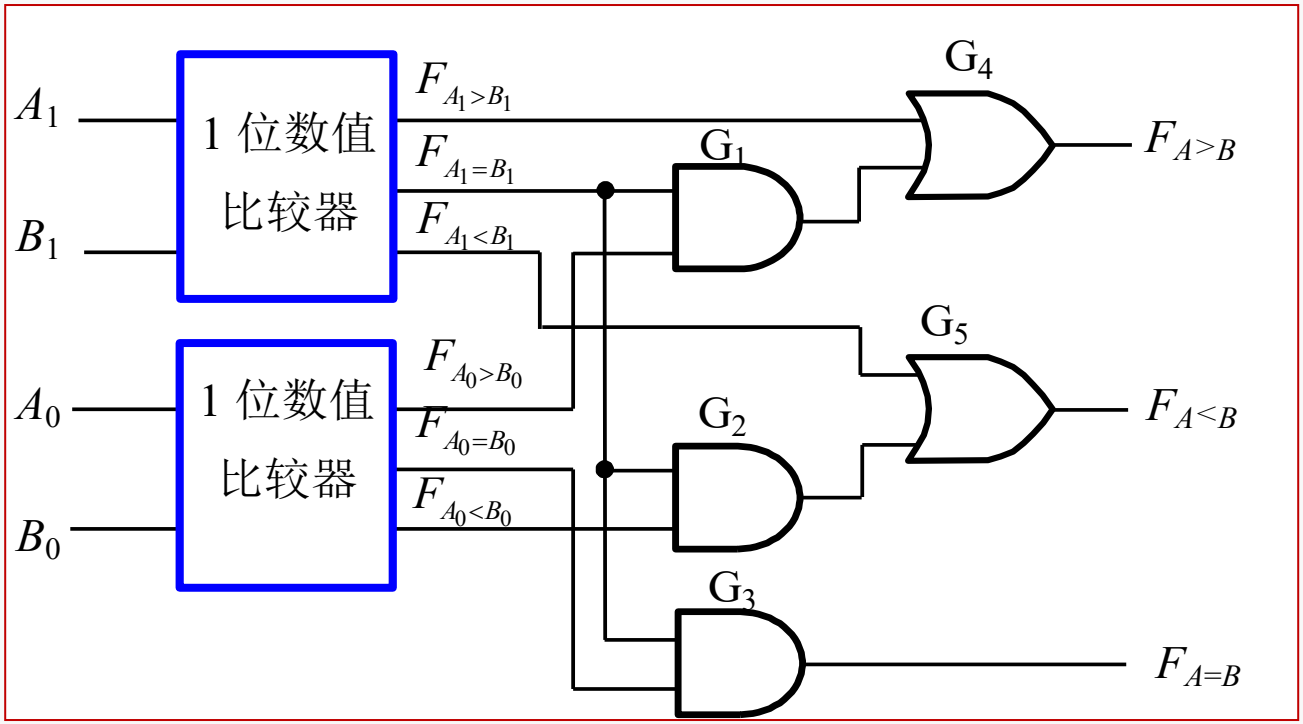
$$F_{A=B} = (A_1=B_1)(A_0=B_0)$$

4.5.4 数值比较器——2位数值比较器

$$F_{A>B} = (A_1>B_1) + (A_1=B_1)(A_0>B_0) \qquad F_{A=B}=(A_1=B_1)(A_0=B_0)$$

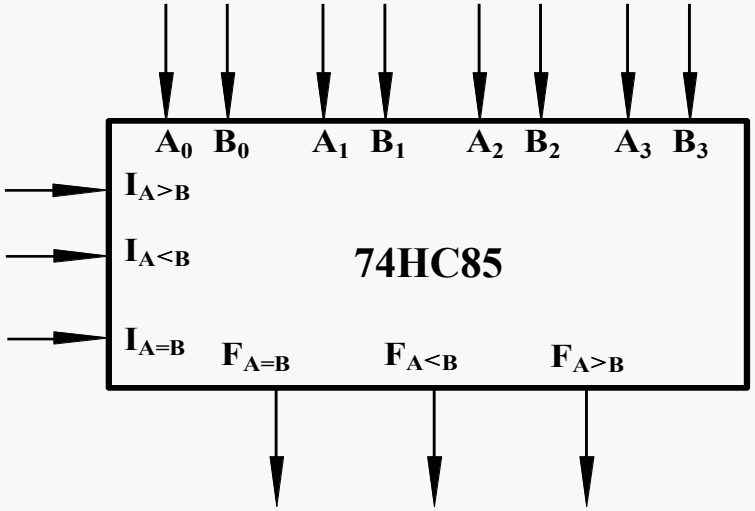
$$F_{A<B} = (A_1<B_1) + (A_1=B_1)(A_0<B_0)$$

2位数值比较器逻辑图



4.5.4 数值比较器——4位数值比较器74HC85

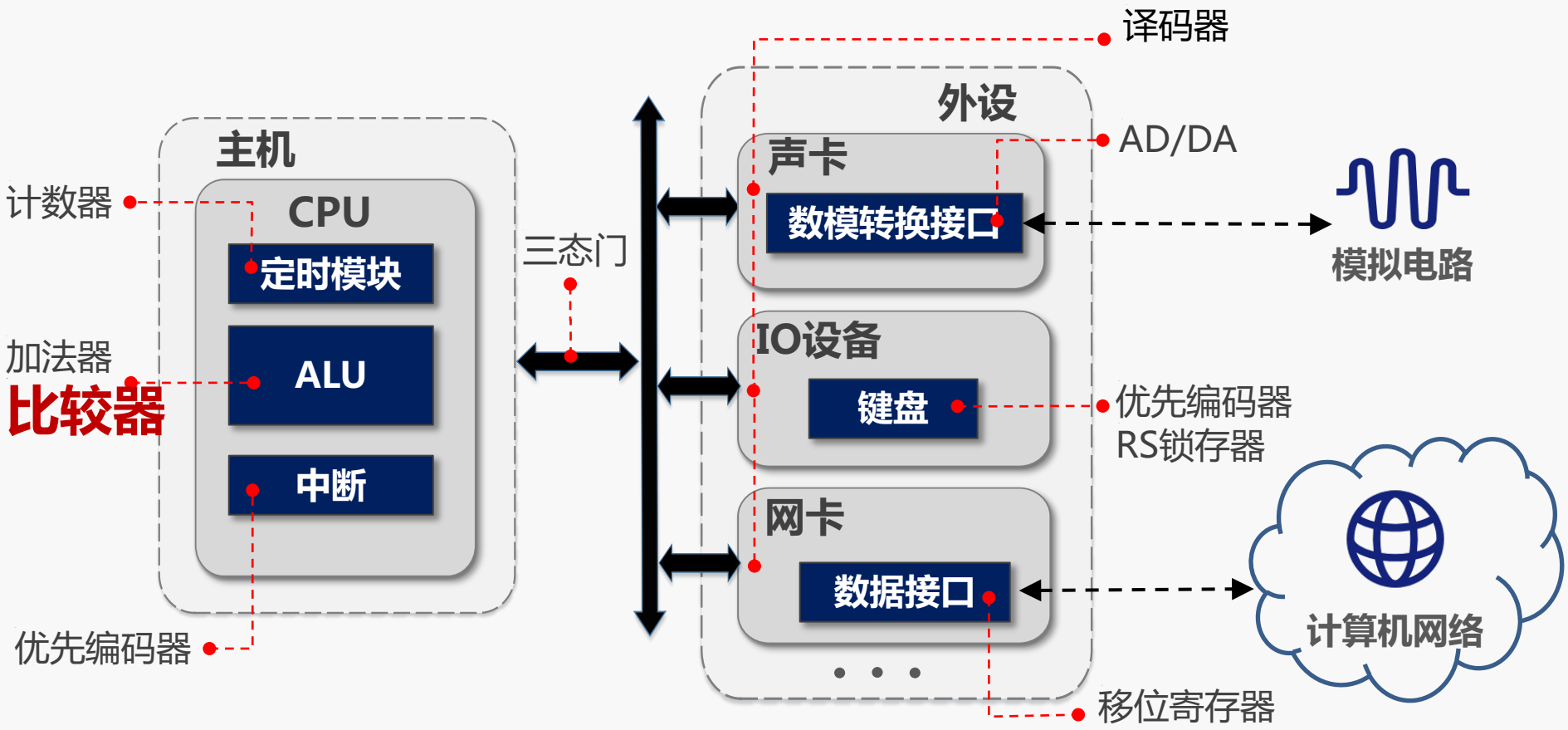
74HC85的示意框图



4位数值比较器74HC85的功能表

输入							输出		
$A_3\ B_3$	$A_2\ B_2$	$A_1\ B_1$	$A_0\ B_0$	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_3 > B_3$	×	×	×	×	×	×	H	L	L
$A_3 < B_3$	×	×	×	×	×	×	L	H	L
$A_3 = B_3$	$A_2 > B_2$	×	×	×	×	×	H	L	L
$A_3 = B_3$	$A_2 < B_2$	×	×	×	×	×	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	×	×	×	×	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	×	×	×	×	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	×	×	×	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	×	×	×	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	L	L	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	H	L	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	×	×	H	L	L	H
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	H	L	L	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	L	L	H	H	L

4.5.4 数值比较器的应用

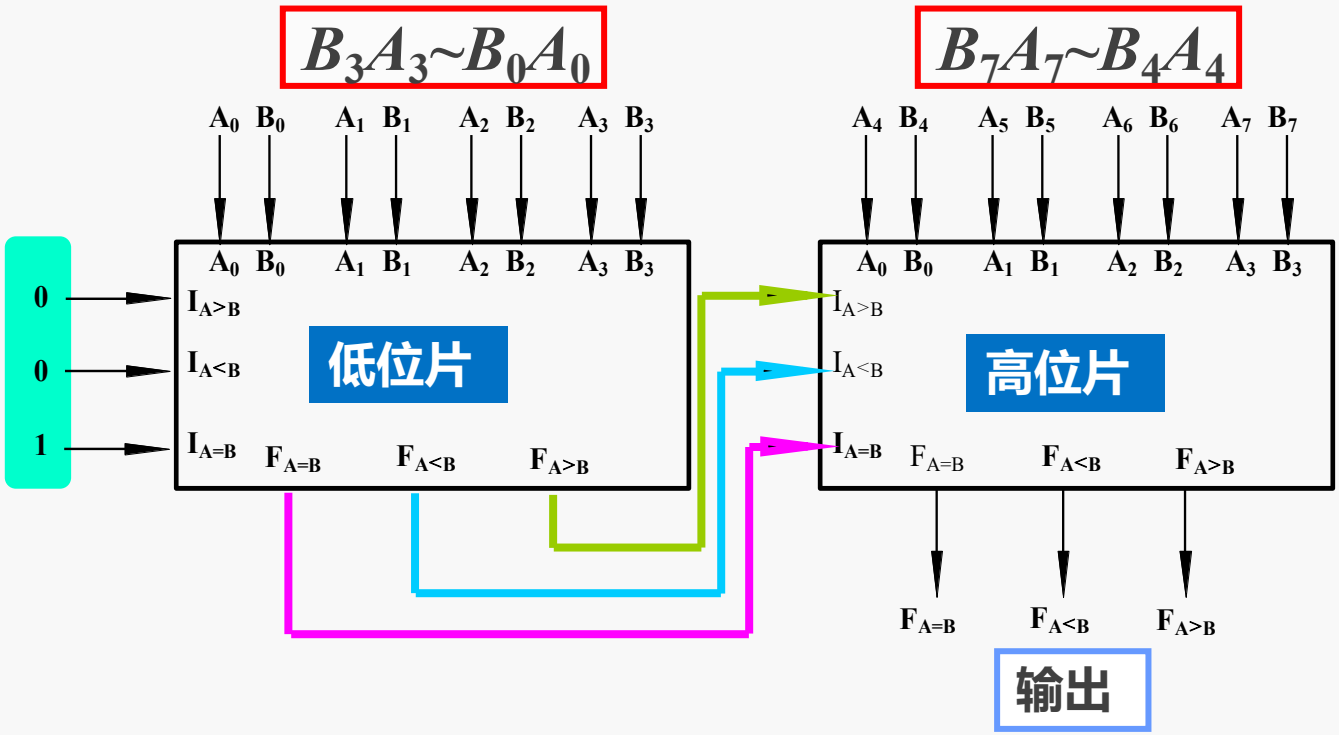


4.5.4 数值比较器——扩展

用两片74HC85组成8位数值比较器

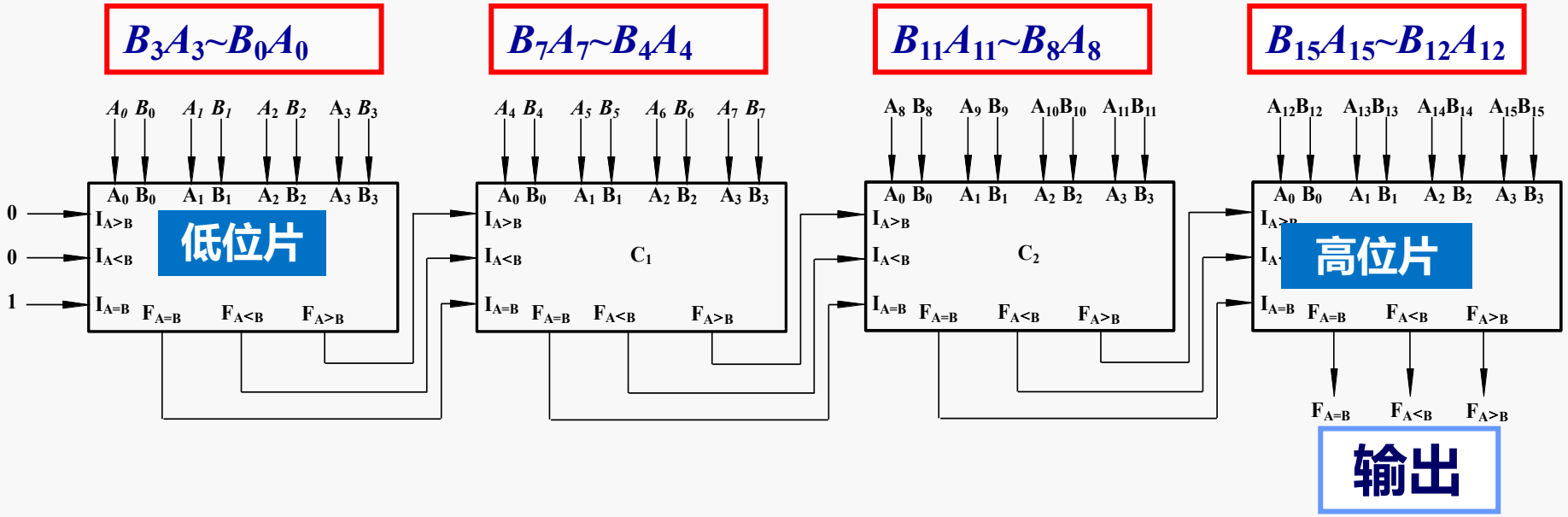
输入：$A=A_7A_6A_5A_4A_3A_2A_1A_0$　　$B=B_7B_6B_5B_4B_3B_2B_1B_0$

输出：$F_{A>B}$、$F_{A<B}$、$F_{A=B}$



4.5.4 数值比较器——扩展

用4片74HC85组成16位数值比较器（串联扩展）

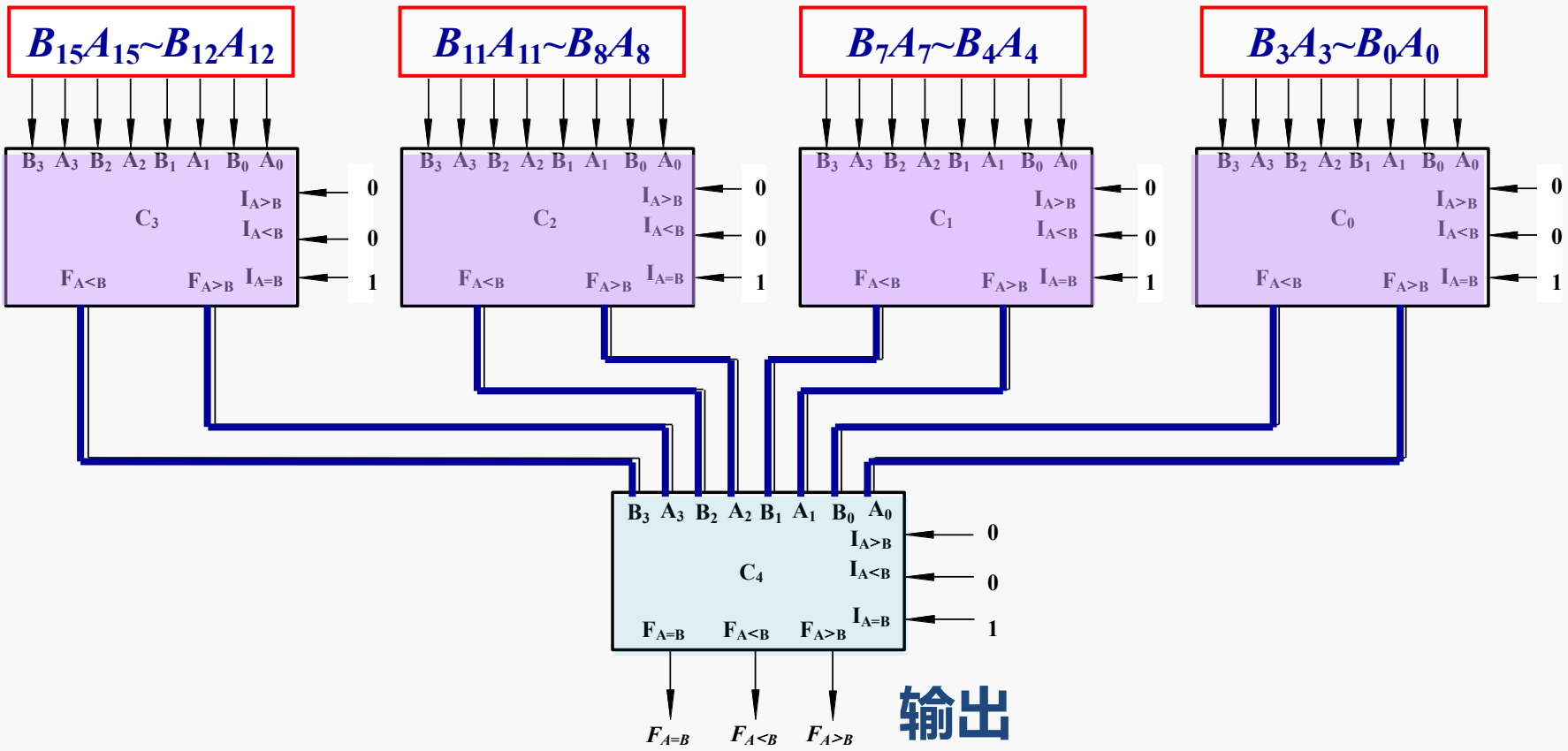


问题：如果每一片延迟时间为10ns，16位串行比较器延迟时间？

串联扩展方式：电路原理**简单**，电路资源消耗**少**，但时延**大**。

4.5.4 数值比较器——扩展

用74HC85组成16位数值比较器（并列扩展）



问题：如果每一片延迟时间为10ns，16位并行比较器延迟时间？

并联扩展方式：电路原理复杂，电路资源消耗多，但时延小。

4.5.4 数值比较器——Verilog HDL描述

```
module Comp(A , B , Fg, Fl , Fe);
parameter D_Width = 4;
input [D_Width-1:0] A, B;
output Fg, Fl , Fe;
reg Fg, Fl , Fe;

always @ (A or B)
if(A>B) begin
    Fg=1'b1; Fl=1'b0; Fe=1'b0;  end
else if (A<B) begin
    Fg=1'b0; Fl=1'b1; Fe=1'b0;  end
else begin
    Fg=1'b0; Fl=1'b0; Fe=1'b1;  end

endmodule
```

问题：如用Verilog设计16位数值比较器时，将会用哪种扩展方式呢？？？

需要用Verilog HDL引导电路所采用的结构。

4.5 常用组合逻辑电路

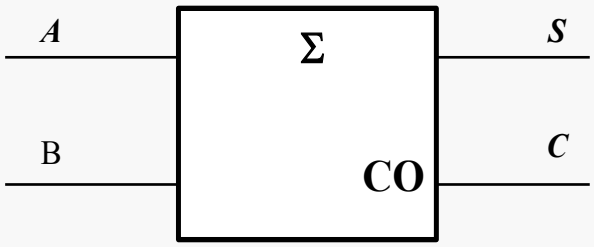
4.5.5 加法器——1位加法器

定义：两个1bit的数相加

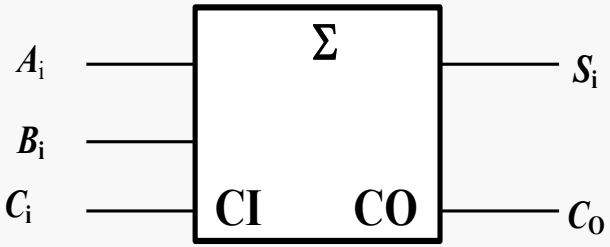
半加器：两个1位二进制数相加，**不考虑**低位进位

全加器：两个1位二进制数相加，且**考虑**低位进位

半加器



全加器

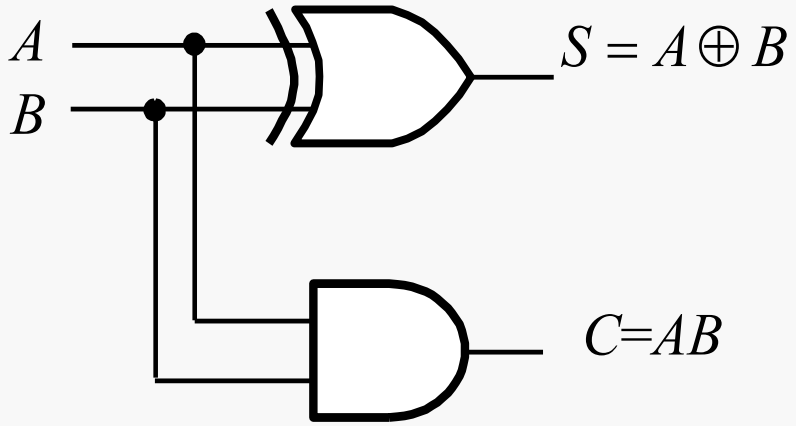


4.5.5 加法器——1位加法器（半加器Half Adder）

半加器的真值表

A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

逻辑图



逻辑表达式 $S = \overline{A}B + A\overline{B}$

$$C = AB$$

如用与非门实现最少要几个门?

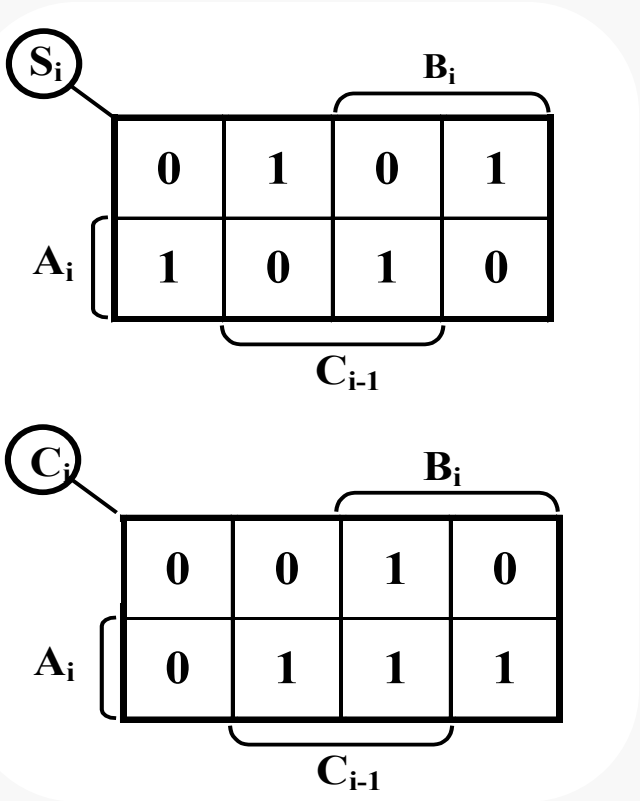
4.5.5 加法器——1位加法器（全加器Full Adder）

全加器进行加数、被加数和低位来的进位信号相加，并根据求和结果给出进位信号。

全加器真值表

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

逻辑表达式



4.5.5 加法器——1位加法器（全加器Full Adder）

逻辑表达式

$$S_i = \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}\overline{C}_{i-1} + A_iB_iC_{i-1}$$

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

$$C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$$

考虑用译码器74HC138或者选择器74HC151实现全加器

考虑基于半加器实现全加器

$$C_i = A_iB_i + A_i\overline{B}_iC_i + \overline{A}_iB_iC_{i-1}$$

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

卡诺图

S_i

A_i \ B_iC_i	00	01	11	10
0	0	1	0	1
1	1	0	1	0

C_i

A_i \ B_iC_i	00	01	11	10
0	0	0	1	0
1	0	1	1	1

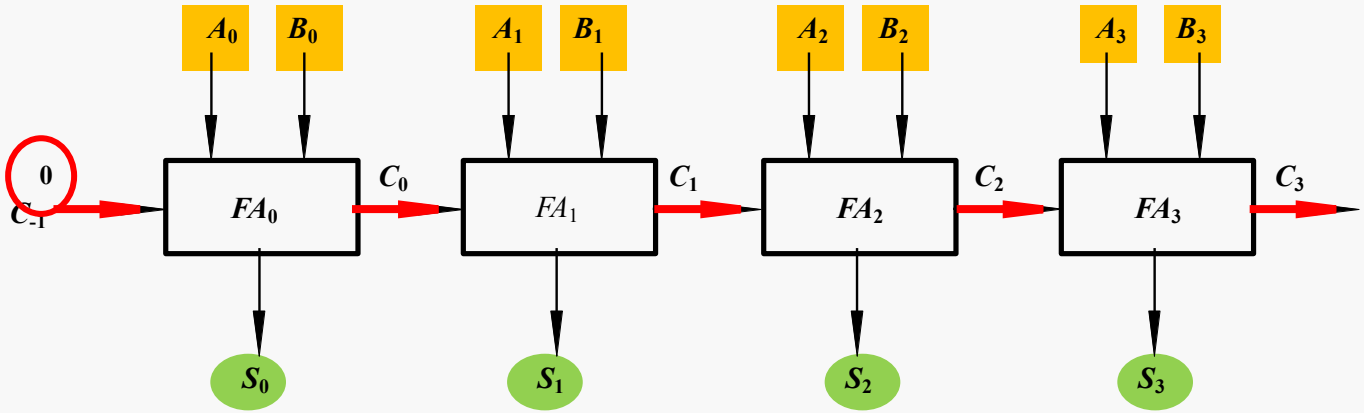
4.5.5 加法器——4位加法器

如何用1位全加器实现两个四位二进制数相加？

$A_3\ A_2\ A_1\ A_0 + B_3\ B_2\ B_1\ B_0 = ?$

$$\begin{array}{r} 1\ 1\ 0\ 1 \\ +\ 1\ 0\ 0\ 1 \\ \hline 1\ 0\ 1\ 1\ 0 \end{array}$$

（1）串行进位加法器



问题：任一位的加法运算必须等待低一位的运算完成之后才能进行，运算速度低。

4.5.5 加法器——4位加法器（超前进位加法器）

提高运算速度的基本思想：设计进位信号产生电路，在输入每位的加数和被加数时，同时获得该位全加的进位信号，而无需等待相邻低位的进位信号。

定义第 i 位的进位信号（C_i）：

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

定义两个中间变量G_i和P_i：

$$G_i = A_iB_i \qquad p_i = A_i \oplus B_i \qquad C_i = G_i + P_iC_{i-1}$$

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

4.5.5 加法器——4位加法器（超前进位加法器）

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

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

$$C_1 = G_1 + P_1 C_0$$

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

$$C_2 = G_2 + P_2 C_1$$

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

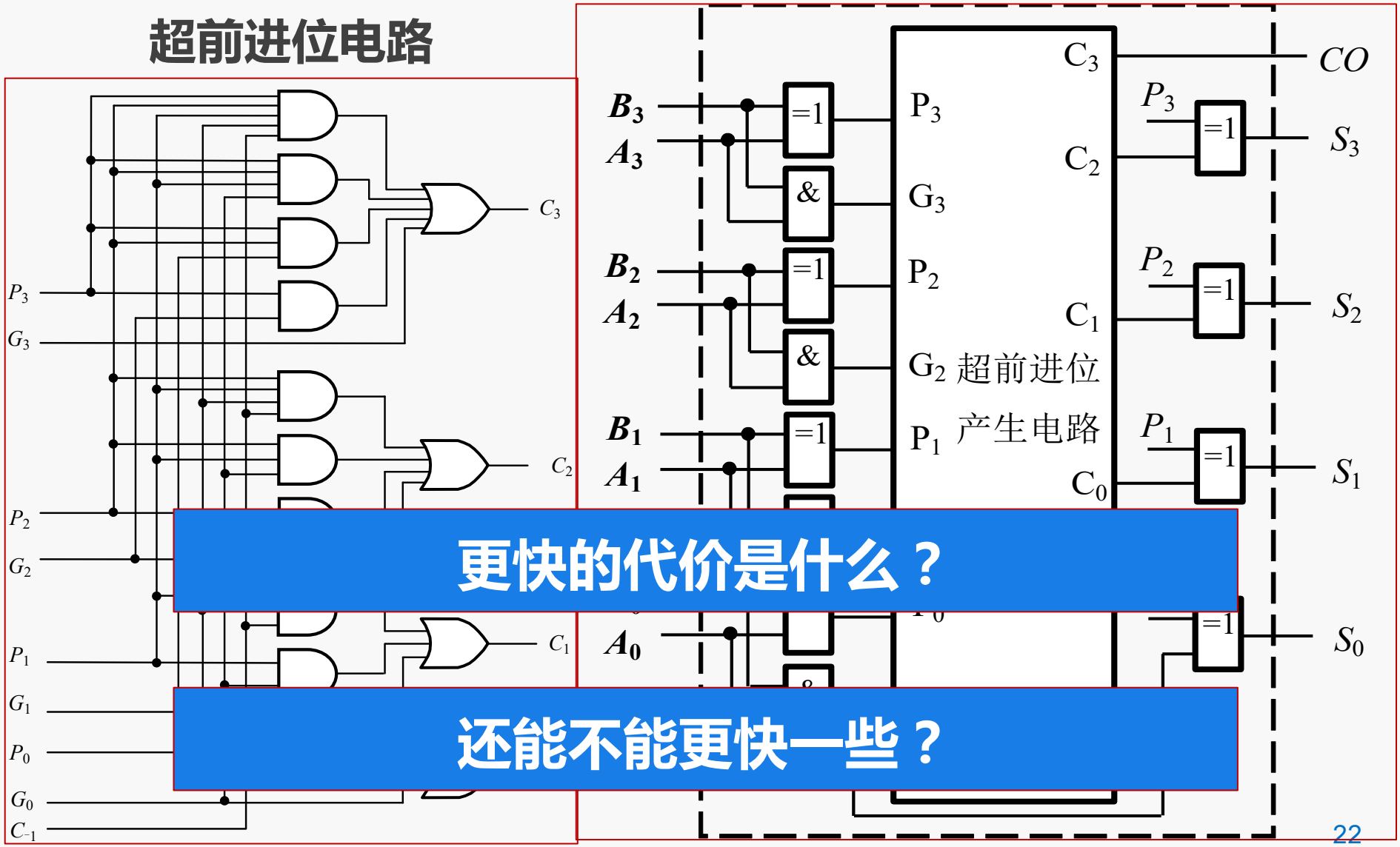
$$C_3 = G_3 + P_3 C_2$$

$$= G_3 + P_3 (G_2 + P_2 C_1) = G_3 + P_3 G_2 + P_3 P_2 C_1$$

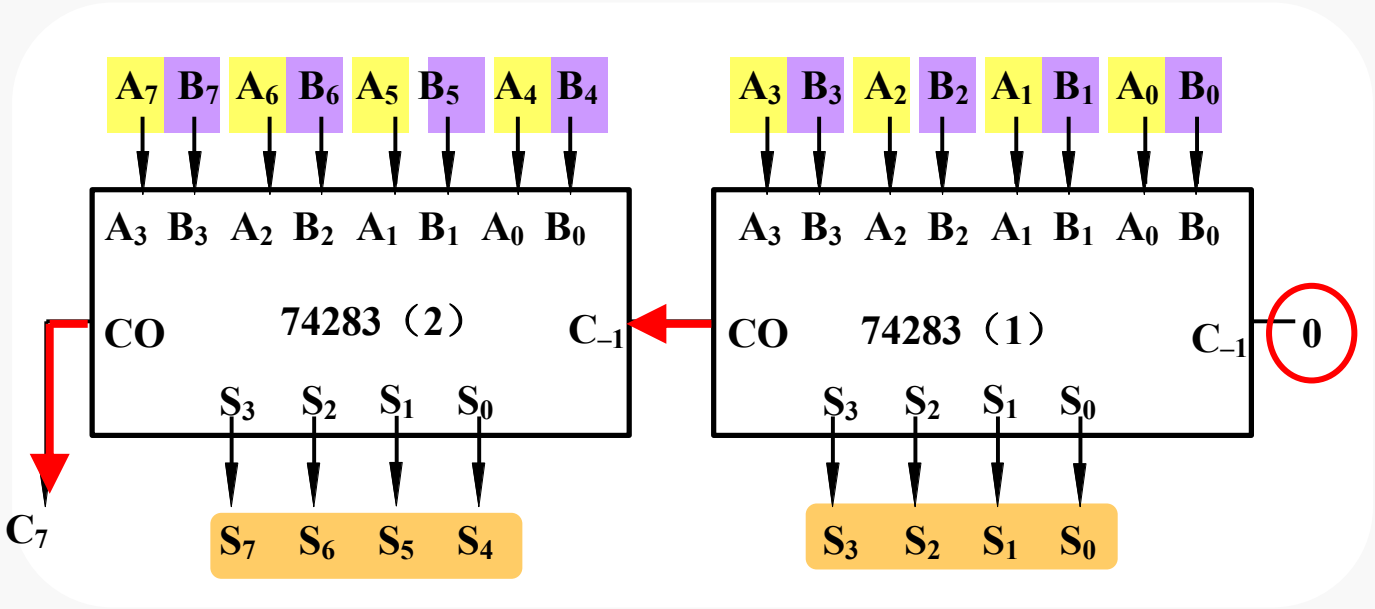
$$= G_3 + P_3 G_2 + P_3 P_2 (G_1 + P_1 C_0)$$

$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 (G_0 + P_0 G_{-1})$$

4.5.5 加法器——4位加法器（超前进位加法器）



4.5.5 加法器——扩展到8位



在片内是超前进位，而片与片之间是串行进位。

时间和空间的折中，是复杂数字电路设计的主题！！！

4.5.5 加法器——Verilog描述

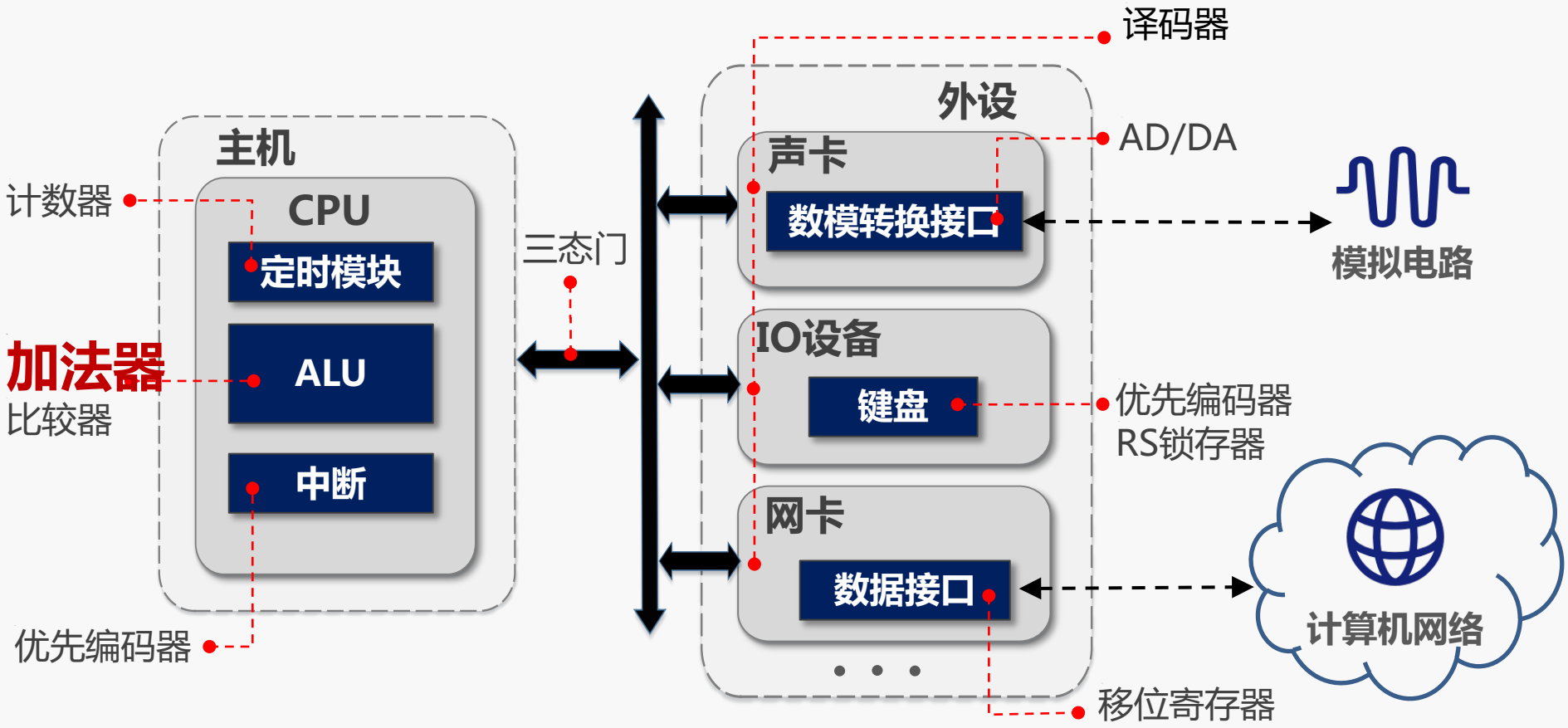
```
module Adder(A , B , S , C);
parameter D_Width = 4;
input [D_Width-1:0] A, B;
output [D_Width-1:0] S;
output C;
reg [D_Width-1:0] S ;
reg C;
endmodule
```

时间和空间的折中

是复杂数字电路设计的主题！！！

问题：用Verilog设计加法器时，用的是串联or并联方式？

4.5.5 加法器的应用


译码器
外设
主机
计数器
CPU
声卡
AD/DA
数模转换接口
模拟电路
定时模块
三态门
加法器
ALU
比较器
IO设备
键盘
优先编码器
RS锁存器
中断
网卡
数据接口
计算机网络
优先编码器
移位寄存器