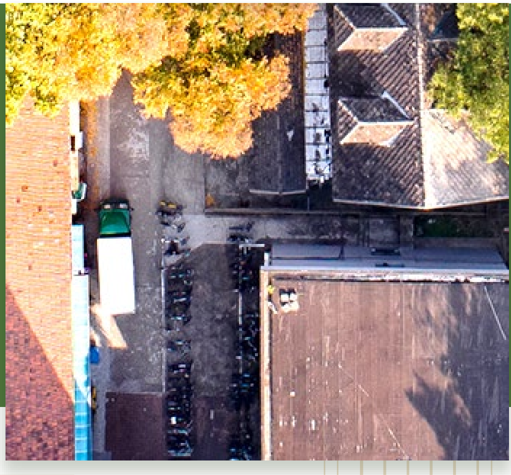
NANJING UNIVERSITY

习题课

Introduction to Computing Systems



南京大學
NANJING UNIVERSITY

01 数据的机器级表示

6.2 回答如下问题：

1）对于 8 位二进制原码、反码和补码整数类型，能够表示的最大的正数分别是多少？请分别以二进制和十进制形式写出结果；

2）对于 8 位二进制原码、反码和补码整数类型，能够表示的最小的负数分别是多少？请分别以二进制和十进制形式写出结果；

3）对于 n 位二进制原码、反码和补码整数类型，能够表示的最大的正数分别是多少？

4）对于 n 位二进制原码、反码和补码整数类型，能够表示的最小的负数分别是多少？

	原码		反码		补码	
8 位最大正数	0111 1111	127	0111 1111	127	0111 1111	127
8 位最小负数	1111 1111	-127	1000 0000	-127	1000 0000	-128
n 位最大正数	$2^{n-1}-1$		$2^{n-1}-1$		$2^{n-1}-1$	
n 位最小负数	$-2^{n-1}+1$		$-2^{n-1}+1$		-2^{n-1}	

6.3 将下列二进制数转化为十进制数，假设此二进制数分别为原码、反码和补码整数。

1）0111

2）1110

3）11111111

4）10000000

	原码	反码	补码
0111	7	7	7
1110	-6	-1	-2
11111111	-127	-0	-1
10000000	-0	-127	-128

6.4 将下列十进制数转化为 8 位二进制原码、反码和补码整数。

1） -86

2） 85

3） -127

4） 127

	原码	反码	补码
-86	11010110	10101001	10101010
85	01010101	01010101	01010101
-127	11111111	10000000	10000001
127	01111111	01111111	01111111

6.5 如果二进制补码整数最后一位是 0，表明该数是偶数，如果最后两位是 00，则表明该数的什么特点?

该数是 4 的倍数。

6.6 做下列二进制加法运算，给出二进制形式的结果：

1） 1010+0101 = 1111

$$\begin{array}{rr} & 1010 \\ + & 0101 \\ \hline = & 1111 \end{array}$$

2） 0001+1111 = 0000

$$\begin{array}{rr} & 0001 \\ + & 1111 \\ \hline = & (1)0000 \end{array}$$

3） 1110+0001 = 1111

$$\begin{array}{rr} & 1110 \\ + & 0001 \\ \hline = & 1111 \end{array}$$

4） 0111+0110 = 1101

$$\begin{array}{rr} & 0111 \\ + & 0110 \\ \hline = & 1101 \end{array}$$

6.7 对于一个二进制数，如果向右移一位，则意味着进行了什么运算?

n/2

第一次作业

6.8 做下列二进制补码整数加法运算，给出十进制形式的结果，并判断是否产生溢出

1)

		1	1	1	1	1	1	0	1
+		0	1	0	1	0	1	0	1
=	(1)	0	1	0	1	0	0	1	0
=	82								

2)

	0	1	1	1
+	0	1	0	1
=	1	1	0	0
=	4			

3)

		1	1	1	1	1	1	1	1
+		0	0	0	0	0	0	0	1
=	(1)	0	0	0	0	0	0	0	0
=	0								

4)

	0	0	0	1
+	1	1	1	0
=	1	1	1	1
=	-1			

补码加法的溢出检查

- 如果两个数符号相同，和的符号不同
- 一个负数和一个正数的和永远不会出现溢出

符号扩展：正数补0，负数补1

第一次作业

6.8 做下列二进制补码整数加法运算，给出十进制形式的结果，并判断是否产生溢出

5)

	0	1	1	1
+	0	0	0	1
=	1	0	0	0
=	-8			

6)

		1	0	0	0
+		1	1	1	1
=	(1)	0	1	1	1
=	7				

7)

		1	1	1	1	1	1	0	0
+		0	0	1	1	0	0	1	1
=	(1)	0	0	1	0	1	1	1	1
=	47								

8)

		1	0	1	0
+		1	1	0	1
=	1	0	1	1	1
=	7				

补码加法的溢出检查

- 如果两个数符号相同，和的符号不同
- 一个负数和一个正数的和永远不会出现溢出

符号扩展：正数补0，负数补1

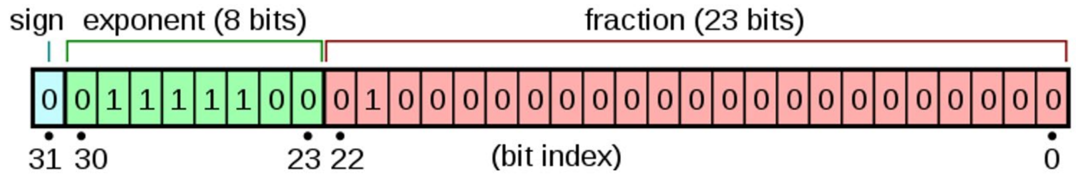
6.10 请给出下列十进制数的 IEEE 754 32 位浮点数表示，结果以十六进制表示

1） 32.9375

2） $-32\frac{45}{128}$

3） -2^{140}

4） 65536



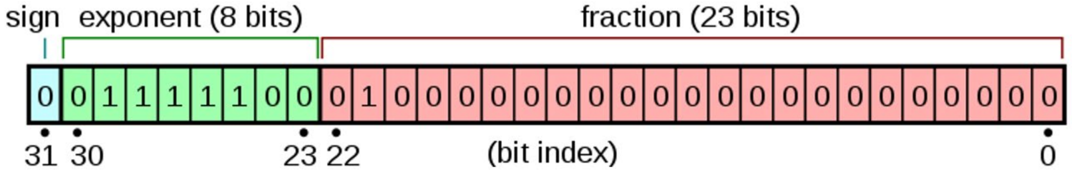
$$\begin{cases} (-1)^{sign} \times 1.fraction \times 2^{E-127}, & 1 \le E \le 254 \\ (-1)^{sign} \times 0.fraction \times 2^{-126}, & E=0 \\ \pm \text{Infinity}, & E=255,\ fraction=0 \\ \text{NaN(not a number)}, & E=255,\ fraction \ne 0 \end{cases}$$

十进制	规格化	浮点数	十六进制H
32.9375	$1.00000\,1111 \times 2^5$	0 10000100 0000 0111 1000 0000 0000 000	4203 C000
$-32\frac{45}{128}$	$-1.00000\,0101101 \times 2^5$	1 10000100 0000 0010 1101 0000 0000 000	C201 6800
-2^{-140}	-1.0×2^{-140}	1 00000000 0000 0000 0000 0100 0000 000	8000 0200
65536	1.0×2^{16}	0 10001111 0000 0000 0000 0000 0000 000	4780 0000

第一次作业

6.11 请给出下列 IEEE 浮点数的十进制数表示

1）0 00000001 00000000000000000000000
2）0 00000000 00000000010000000000000
3）1 11111011 00000000000000000000000
4）1 10000001 10101000000000000000000
5）0 01111101 01010100000000000000000



$$\begin{cases} (-1)^{sign} \times 1.fraction \times 2^{E-127}, & 1 \le E \le 254 \\ (-1)^{sign} \times 0.fraction \times 2^{-126}, & E=0 \\ \pm\text{Infinity}, & E=255,\ fraction=0 \\ \text{NaN(not a number)}, & E=255,\ fraction \ne 0 \end{cases}$$

浮点数	十进制表示
0 00000001 00000000000000000000000	2^{-126}
0 00000000 00000000010000000000000	2^{-136}
0 11111011 00000000000000000000000	-2^{-124}
1 10000001 10101000000000000000000	-6.625 或 $-\frac{53}{8}$
0 01111101 01010100000000000000000	0.33203125 或 $\frac{85}{256}$

6.14 如下代码分别输出哪些内容?

1） printf ("%c\n", 13 + 'A'); ⟹ 13 + 65 = 78 ⟹ N

2） printf ("%x\n", 130); ⟹ 0000 ... 0000 1000 0010 ⟹ 82

答： 1) 输出字母N

2) 输出十六进制数：82

6.15 请解释如下代码段的作用。

```
char nextChar;
int x;
scanf("%c", &nextChar);
printf("%d\n", nextChar);
scanf("%d", &x);
printf("%c\n", x);
```

- 输入一个字符，输出其 ASCII 的数值
- 输入一个整数，输出其作为 ASCII 码代表的字符

附加题1

求满足条件的 $1+2+3+\cdots+n \le 2147483647$ 的最大整数 n，对于如下程序段，请解释：为什么会出现无限循环？

```
int n = 1, sum = 0;
while (sum <= 2147483647) {
  sum += n;
  n++;
}
printf("n=%d\n", n - 1);
```

答：n 的值为 65536 时，计算 sum 溢出，sum 小于 0，则循环条件仍然满足，不会跳出循环

附加题2

使用 $printf(\text{“\%. 16f\textbackslash n”}, 3.14)$; 语句，输出 3.14 的值，为什么输出结果是“3.1400000000000001”，即小数末尾为什么会出现一个1？提示：IEEE 754 64位浮点数标准的分数域为 52 位。

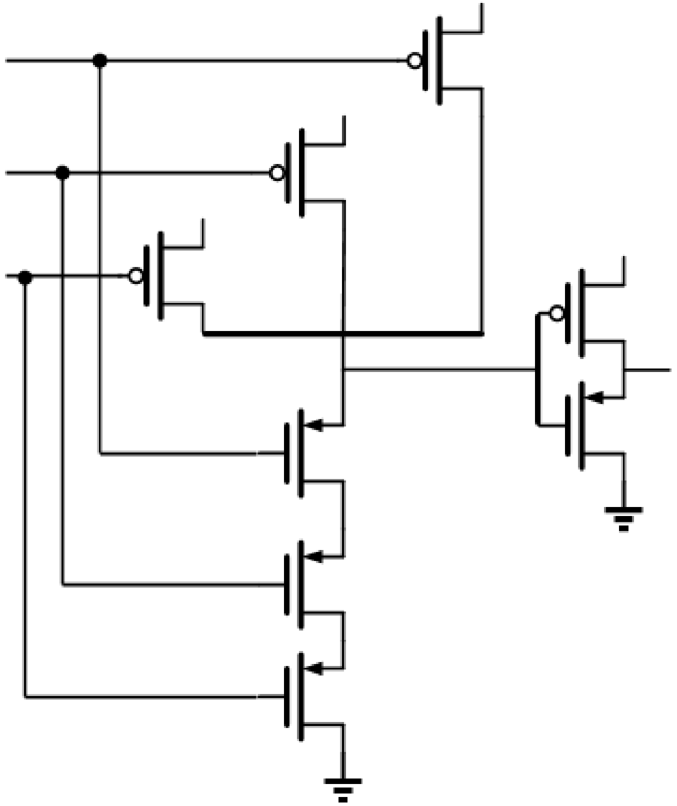
答：3.14 的64位二进制表示为 0 10000000000 1001 0001 1110 1011 1000 0101 0001 1110 1011 1000 0101 0001 1111，与 float 类型相比，拥有更高的精度，但是仍然存在误差。计算到小数点后 16 位的结果是 3.1400000000000001。

南京大学
NANJING UNIVERSITY

02 数字逻辑电路

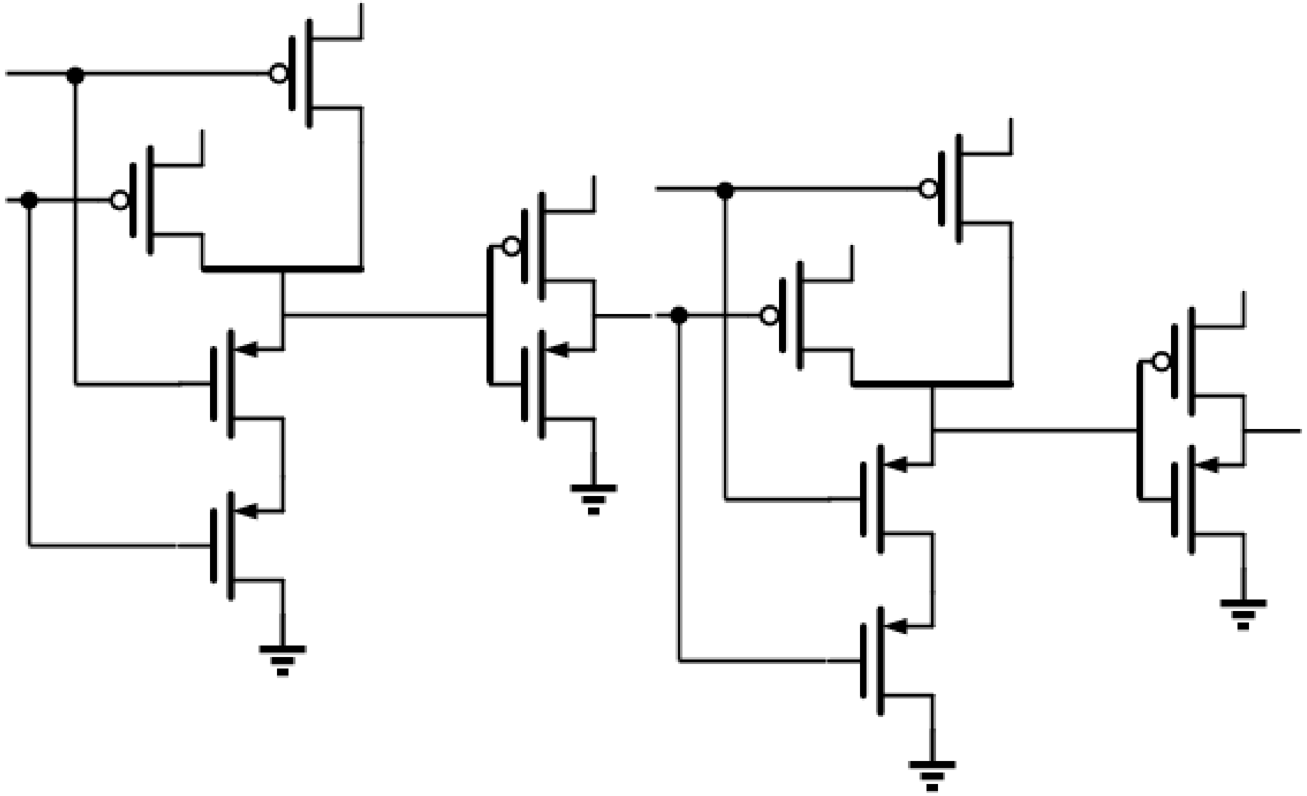
7.1 1）请分别画出三个输入的与门和三个输入的或门的晶体管级电路图。

与门



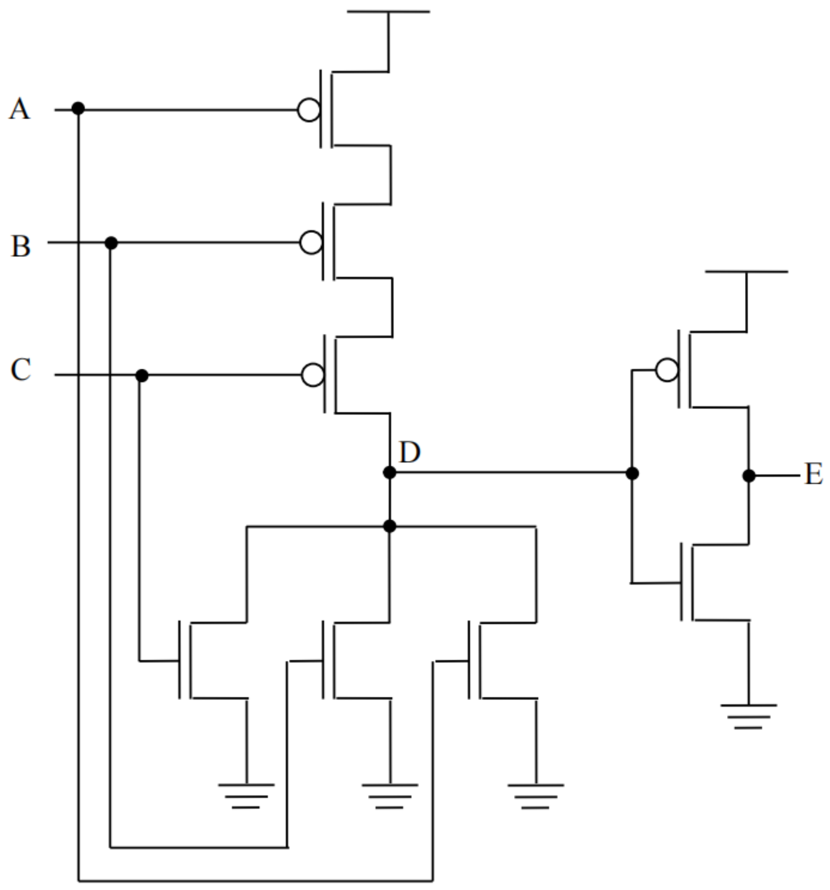
7.1 1）请分别画出三个输入的与门和三个输入的或门的晶体管级电路图。

与门



7.1 1）请分别画出三个输入的与门和三个输入的或门的晶体管级电路图。

或门

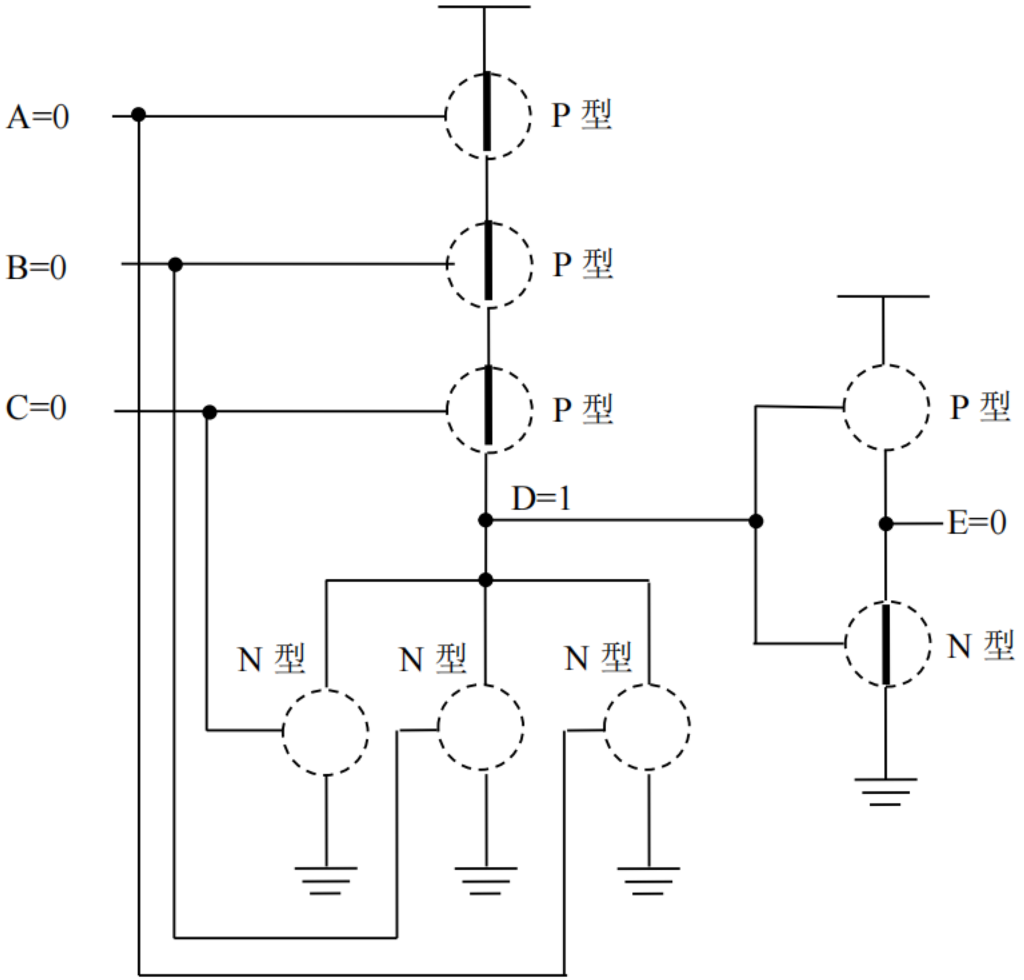


7.1 2）对于如下输入，请分别在其与门和或门晶体管级电路图中标出其表现。

I． A=0, B=0, C=0； II． A=0, B=0, C=1； III． A=1, B=1, C=1

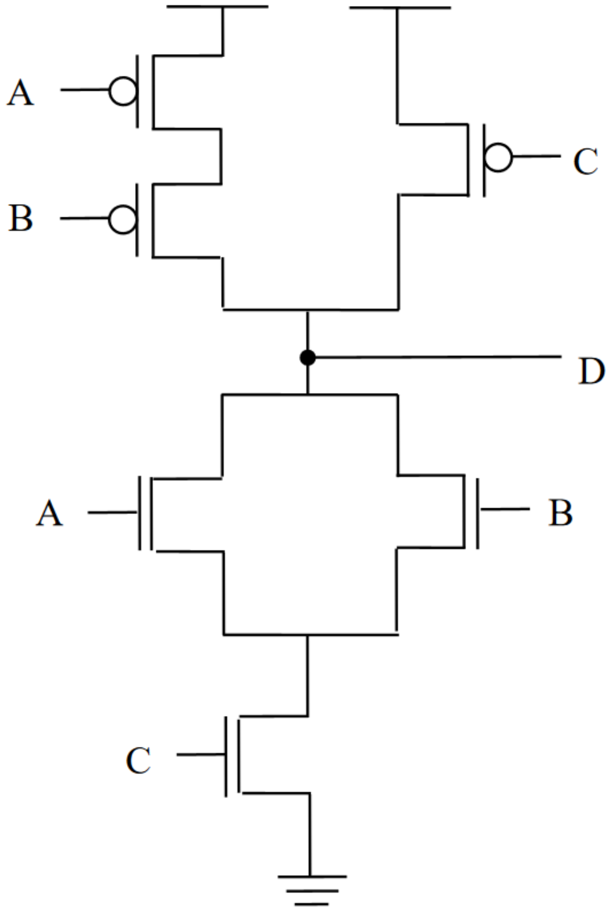
以或门为例

A=0,B=0,C=0



7.2 1）给出下图所示的晶体管级电路的真值表。

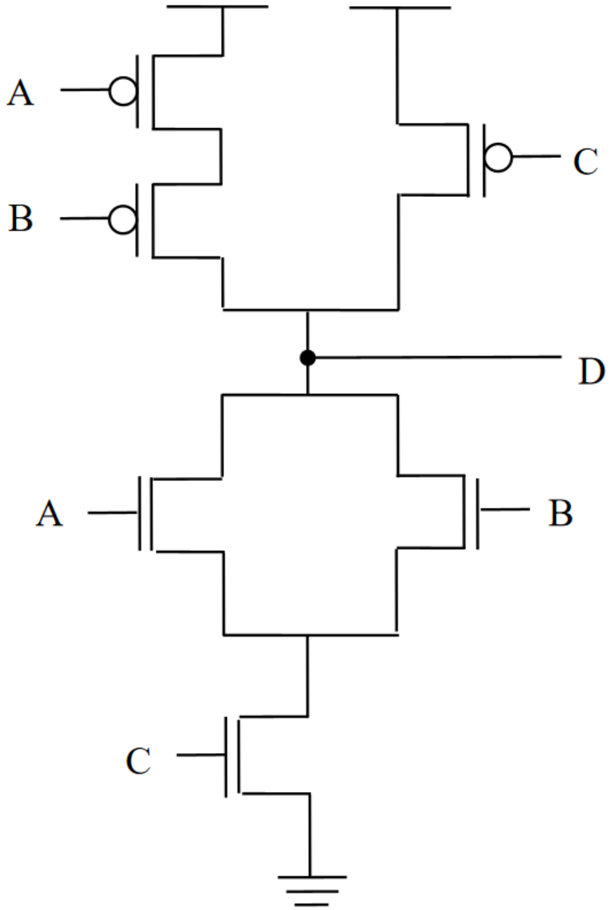
2）使用与、或、非门，给出该真值表的门级电路图。

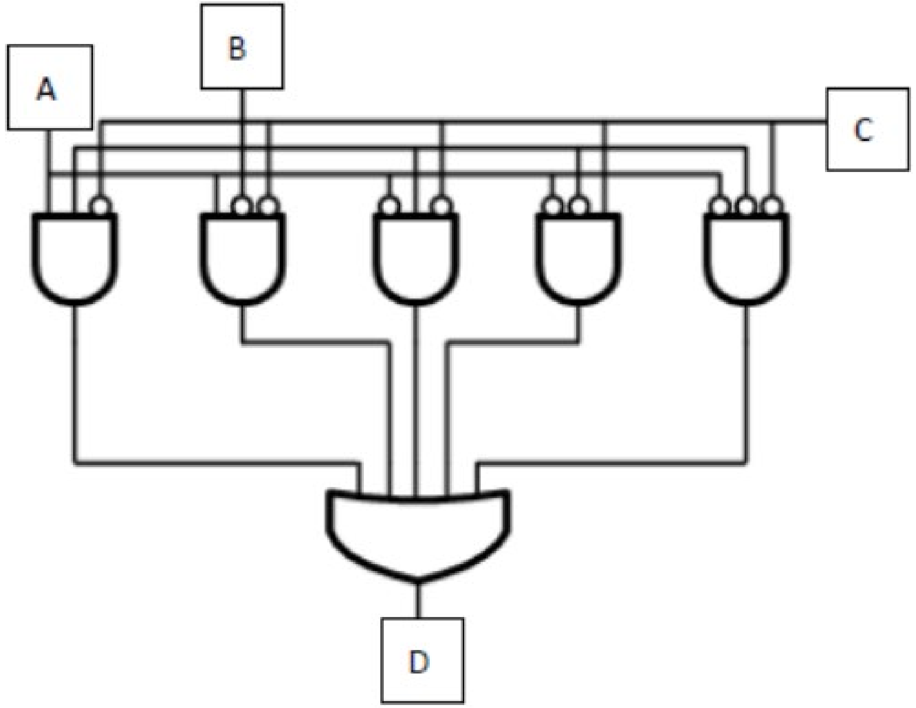


A	B	C	D
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

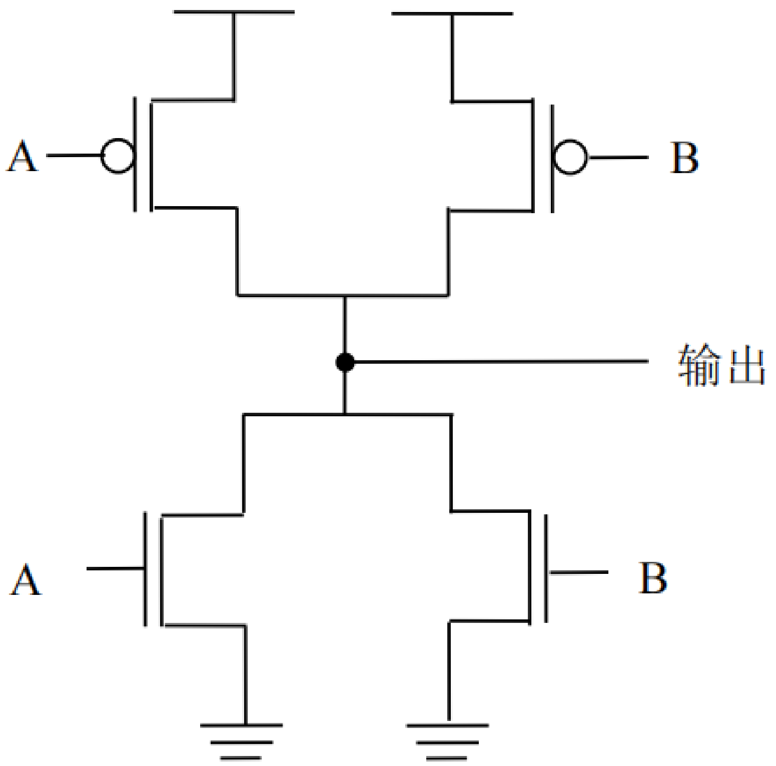
7.2 1）给出下图所示的晶体管级电路的真值表。

2）使用与、或、非门，给出该真值表的门级电路图。



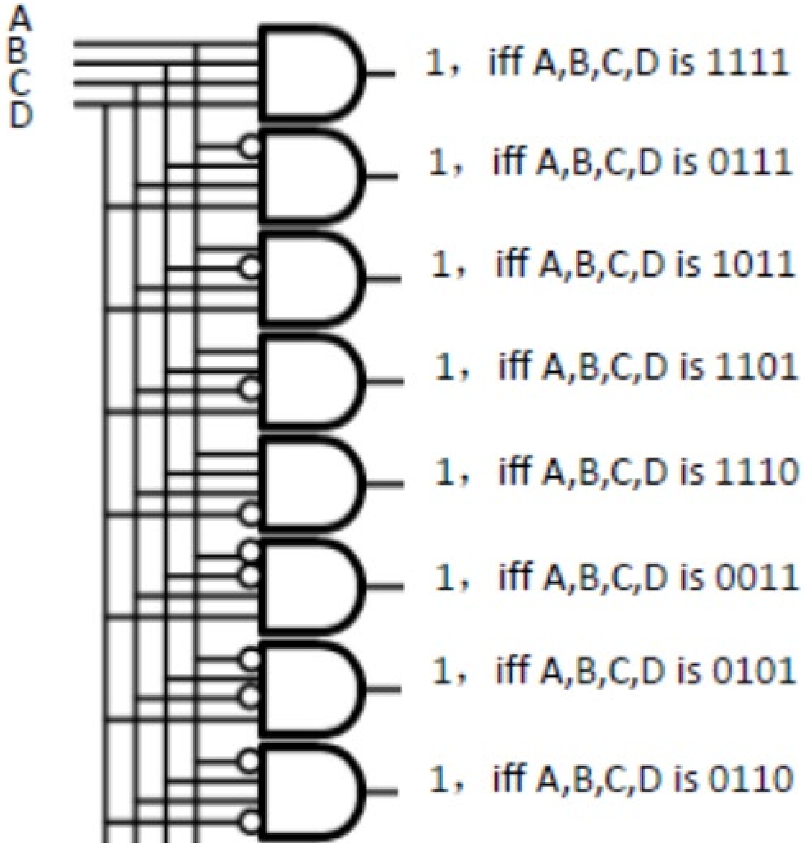


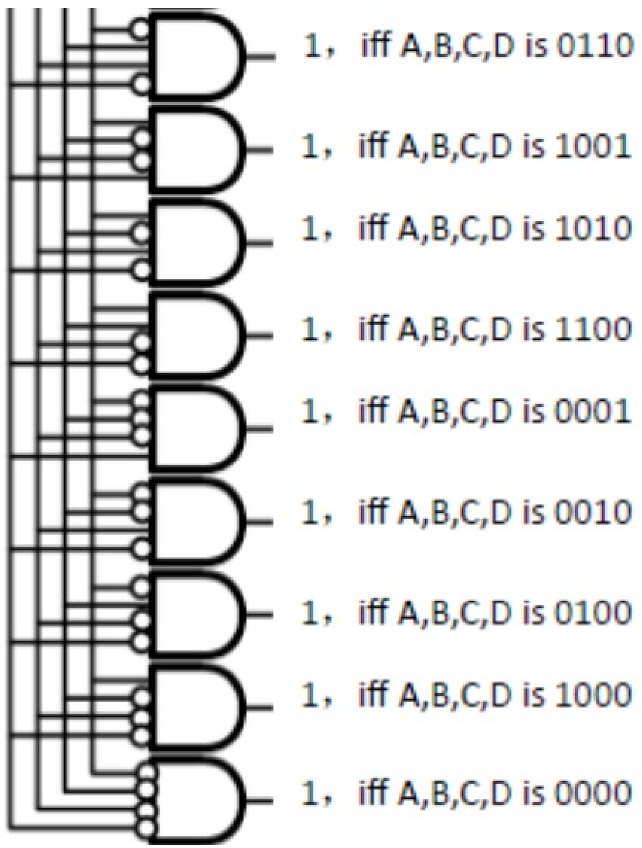
7.3 如下图所示的电路有一个缺陷，请指出该缺陷。



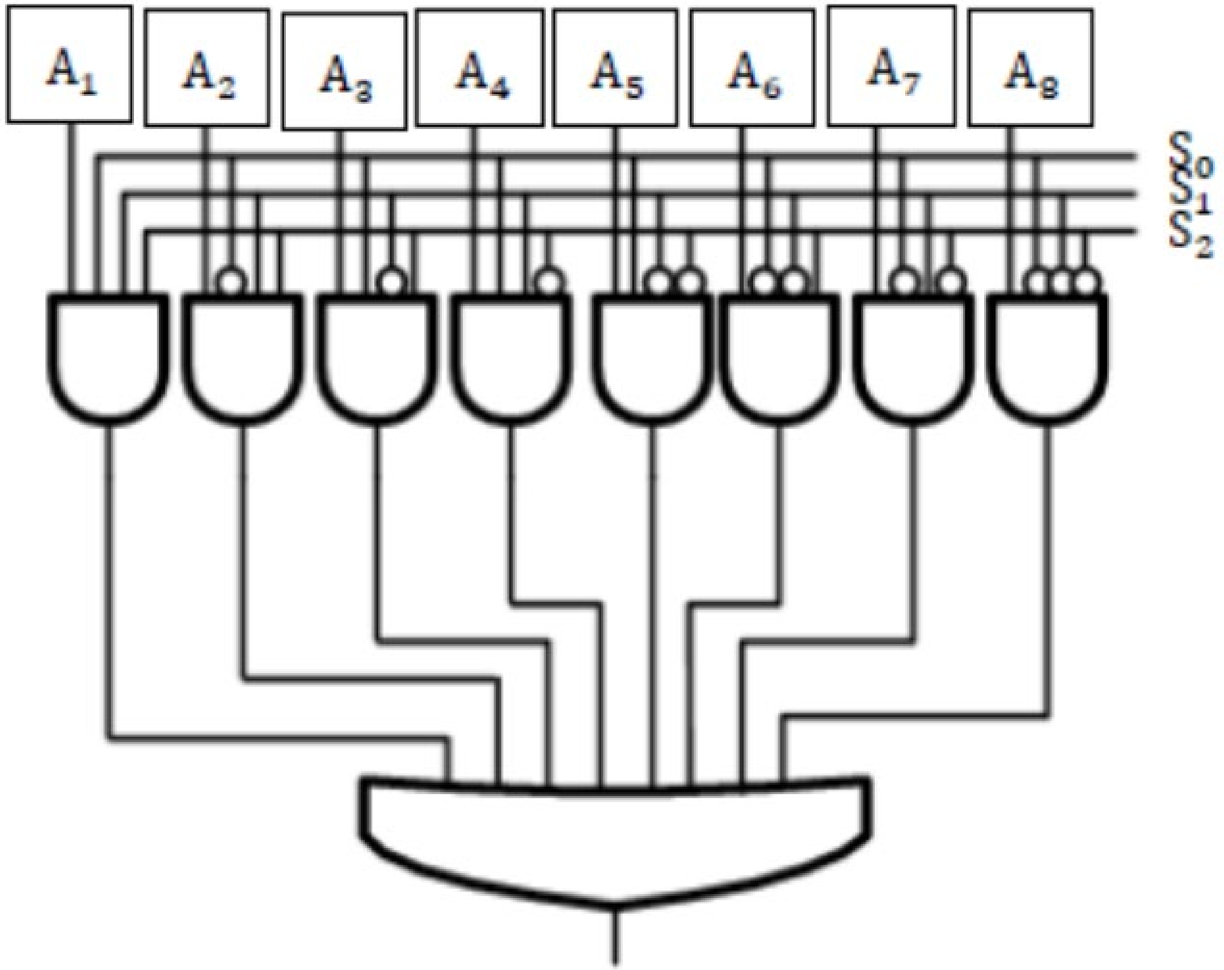
缺陷：若 A=0,B=1 或 A=1,B=0，则同时接到了电源正极和地，出现短路。

7.4 请画出有 4 个输入的译码器的门极电路图，并注明各输出为 1 的条件。

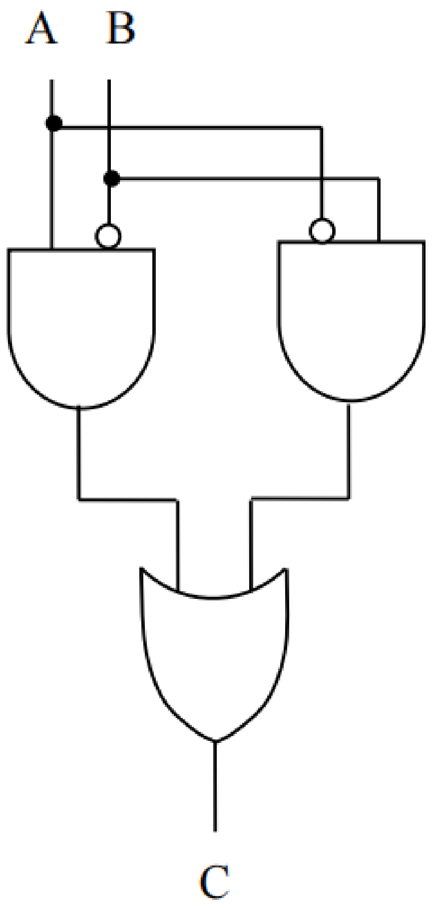




7.5 请画出有 8 个输入的多路选择器的门极电路图。



7.6 使用与、或、非门，给出异或函数的门级电路图。



NANJING UNIVERSITY

第二次作业

7.7 对于如下真值表，请使用 3.4.4 节给出的算法（可编程逻辑阵列），生成其门级逻辑电路。

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$$X = \overline{ABC} + \bar{A}B\bar{C} + A\overline{BC}$$

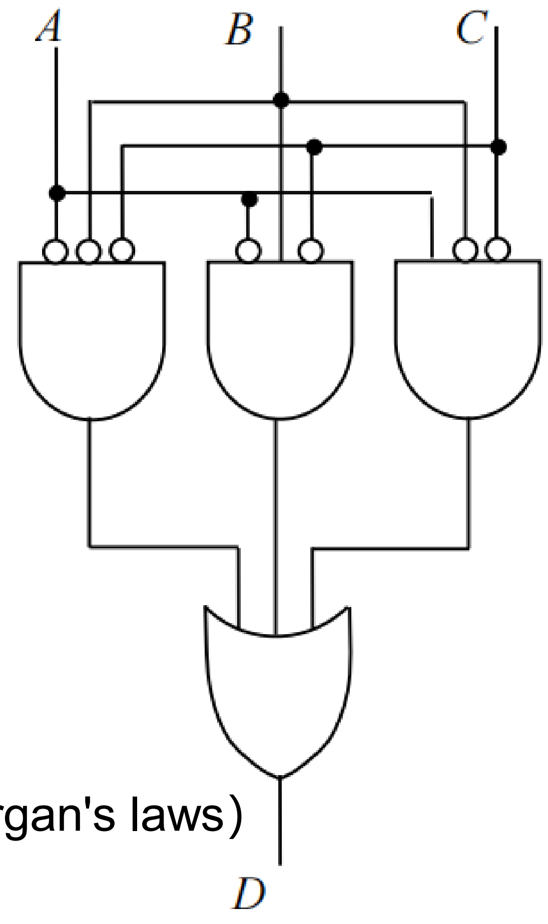
$$= (\overline{AB} + \bar{A}B + A\bar{B})\bar{C}$$

$$= (\bar{A} + A\bar{B})\bar{C}$$ 消去律

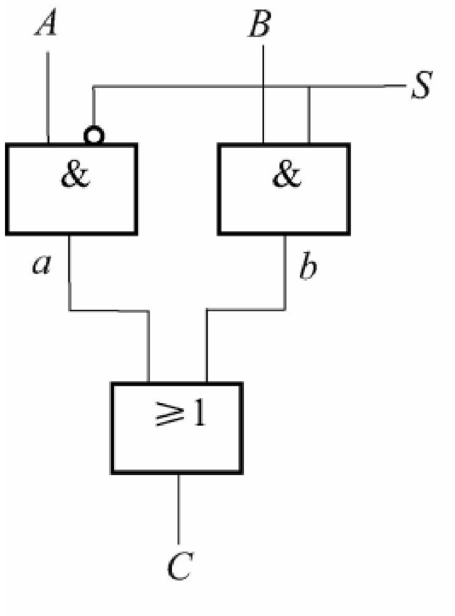
$$= (\bar{A} + \bar{B})\bar{C}$$ 反演律（De Morgan's laws）

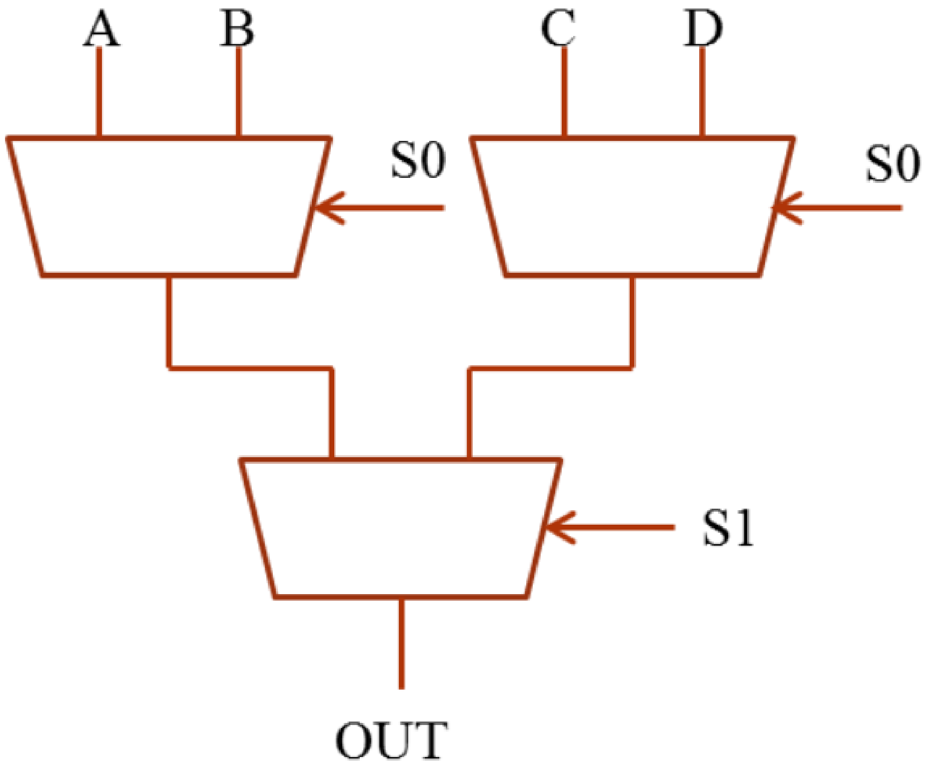
$$= \overline{AB} \cdot \bar{C}$$ 反演律

$$= \overline{AB + C}$$

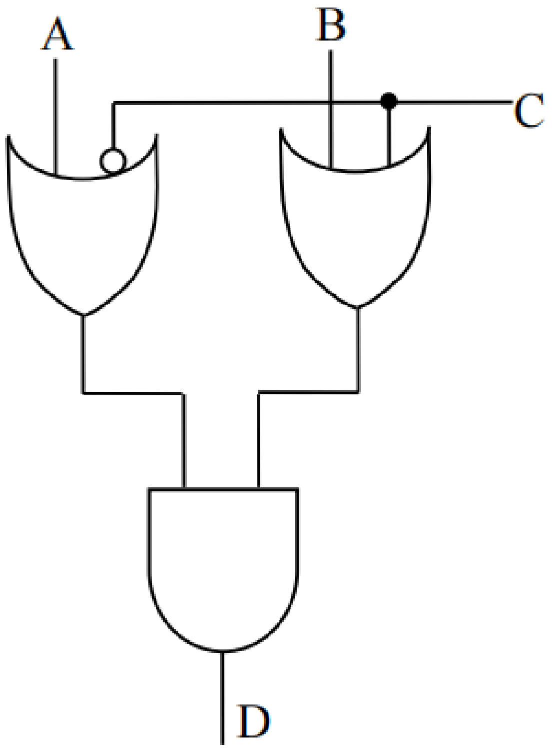


7.8 只使用 2 选 1 的多路选择器，就可以实现 4 选 1 的多路选择器，给出其电路图。



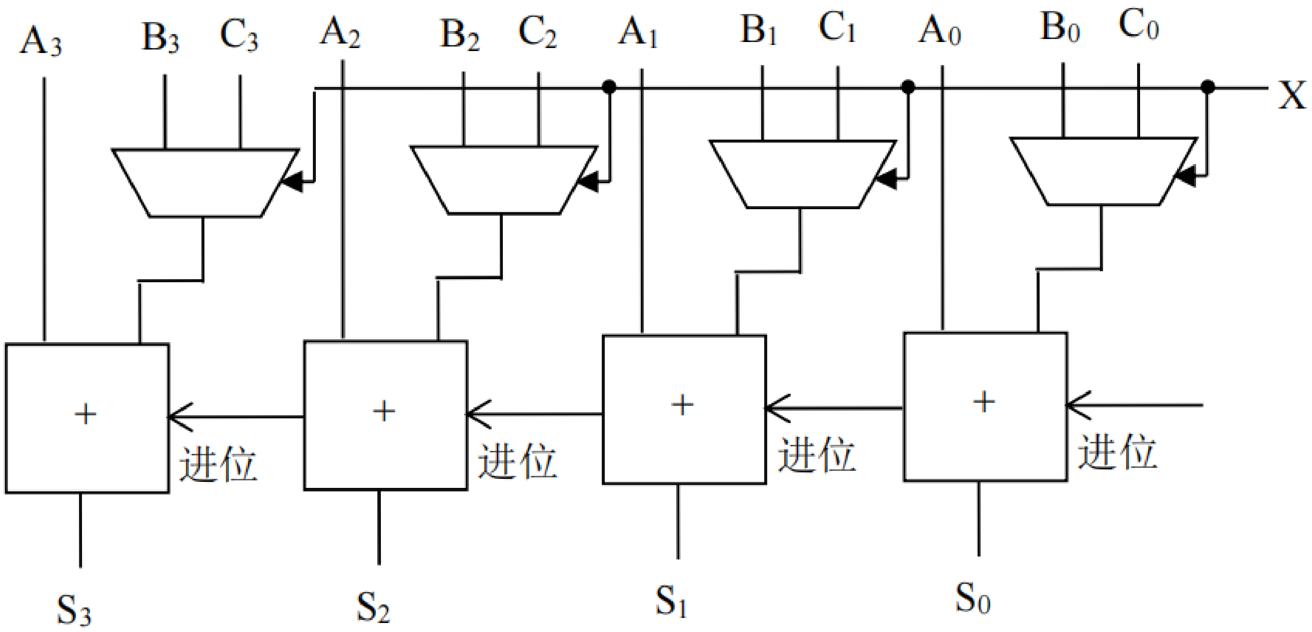


7.9 根据下图所示的逻辑电路图，写出相应的真值表。



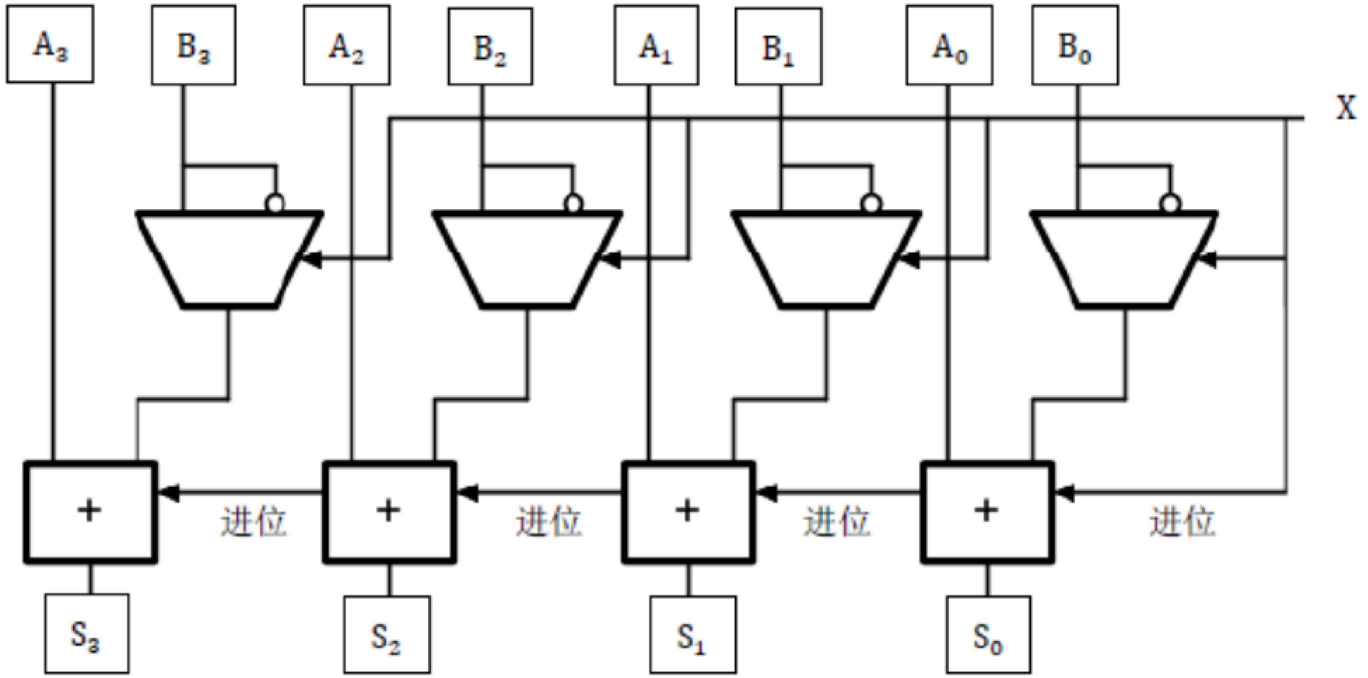
A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

7.10 1）下图中的每个矩形都表示一个全加法器，当 X=0 和 X=1 时，电路的输出分别是什么？



- 当 X=0 时，S = A + B
- 当 X=1 时，S = A + C

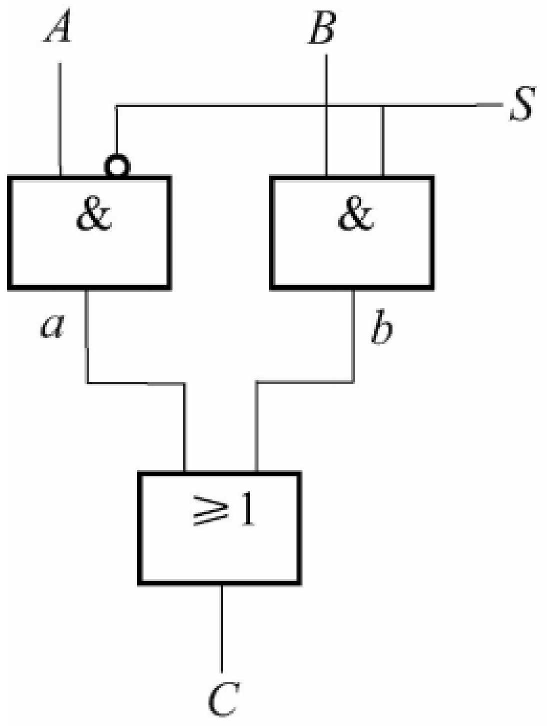
7.10 2）在该电路图的基础上，构建一个可以实现加法/减法运算的逻辑电路图。



- 当 X=0 时，S = A + B
- 当 X=1 时，S = A - B

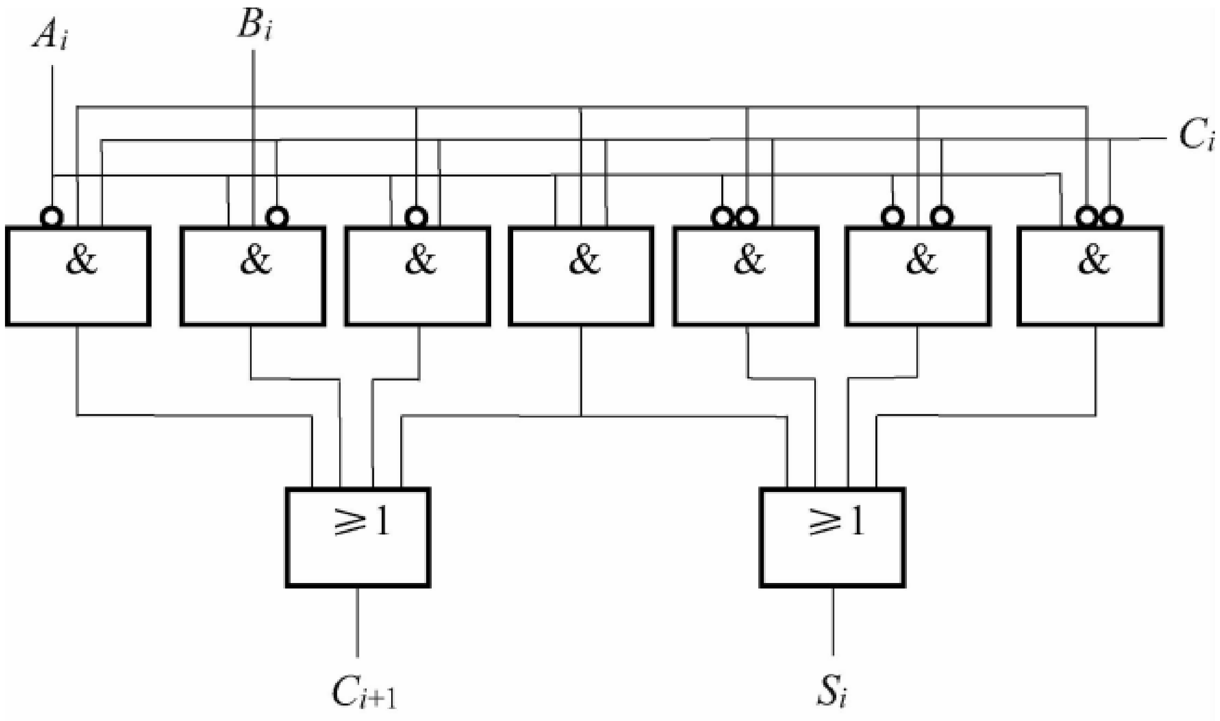
7.11 一个逻辑结构的速度与从输入到达输出，需传递经过的逻辑门的最长路径有关。假设与、或、非门都被计为一个门延迟，例如，两个输入的译码器的传递延迟等于 2（参照图 7.10），这是因为有些输出需经过两个门的传递。

1）两个输入的多路选择器的传递延迟是多少（参照图 7.11）？



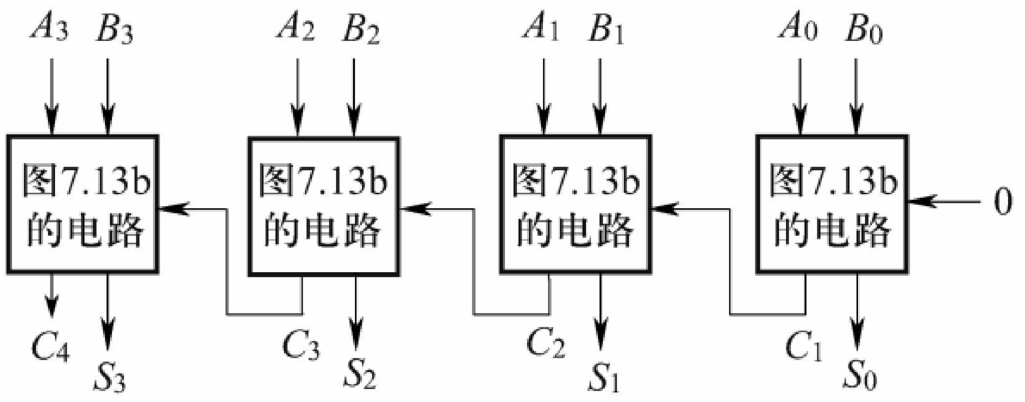
答：3

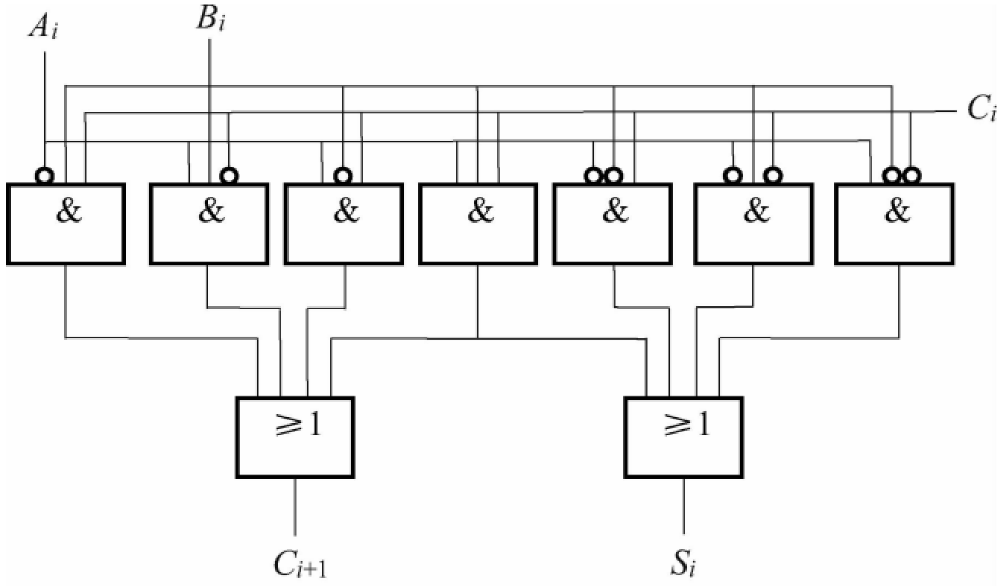
7.11 2）1 位的全加法器的传递延迟是多少（参照图 7.13(b)）？



答：3

7.11 3）4 位的全加法器的传递延迟是多少（参照图 7.13(c)）？





答：4 * 3 = 12

7.11 4）32 位的全加法器的传递延迟是多少?

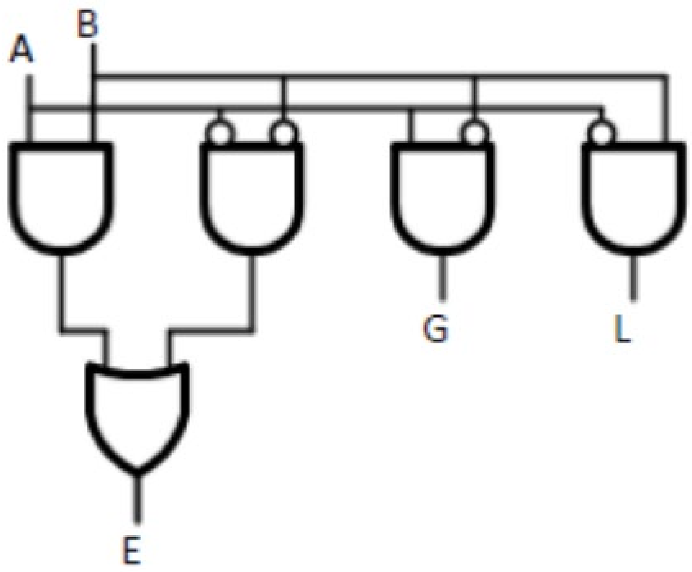
4) 答：32 * 3 = 96

7.12 设计一个 1 位的比较器，该比较器电路有两个 1 位的输入 A 和 B，有 3 个 1 位的输出 G（greater，大于）、E（equal，等于）和 L（less，小于）。当 A > B 时，G 为 1，否则，G 为 0；当 A = B 时，E 为 1，否则，E 为 0；当 A < B 时，L 为 1，否则，L 为 0。

1）给出此 1 位比较器的真值表。

2）使用与、或、非门实现此比较器电路

A	B	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0



7.13 参照下图，回答问题

1）当 S 和 R 都为 0 时，此逻辑电路的输出是什么?

答：A=0/a=0 时 B=1/b=1, A=1/a=1 时 b=0/B=0，

即 a 的值可以为 0，也可以为 1(保持状态)

2）如果 S 从 0 转换到 1，输出是什么?

答：a 的值为 0，b 为 1

3）此逻辑电路是存储元件吗?

答：是

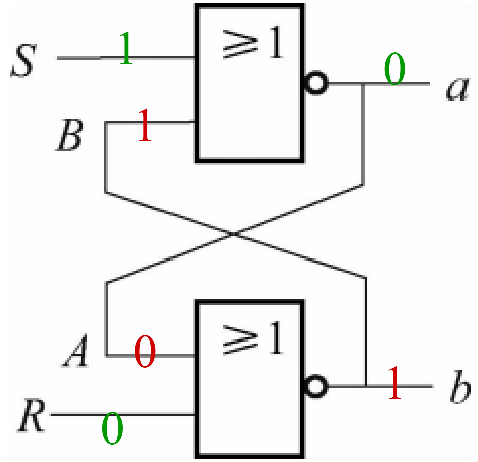


图 7.32　7.13 题电路

7.14 某个计算机有 4 个字节的寻址能力，访问其存储器的一个单元需要 64 位，该存储器的大小是多少（以字节为单位）？此存储器共存储多少位?

答:

存储器大小 = 地址空间(可唯一标识的单元总数) × 寻址能力(存储在每个单元中的位数) $= 2^{64} * 4 = 2^{66}$(byte) $= 2^{69}$(bit)

7.15 8 位被称为一个字节（byte），4 位被称为一个单元组（nibble）。一个字节可寻址的存储器使用 14 位的地址，那么，此存储器共存储了多少单元组?

答：$2^{14} * 8 / 4 = 2^{15}$(nibble)

7.16 对于图 7.18所示的 4×2 位大小的存储器，回答以下问题：

1）如果向单元 3 存储数值，A[1:0]和 WE 必须被设置为什么值？

2）如果将此存储器的单元数目从 4 增长到 10，需要多少条地址线？存储器的寻址能力是否发生变化？

1) A[1:0] 必须被设置为 11，WE 必须被设置为 1

2) 答：需要 4 条地址线，存储器的寻址能力不变

(存储在每个单元中的位数)

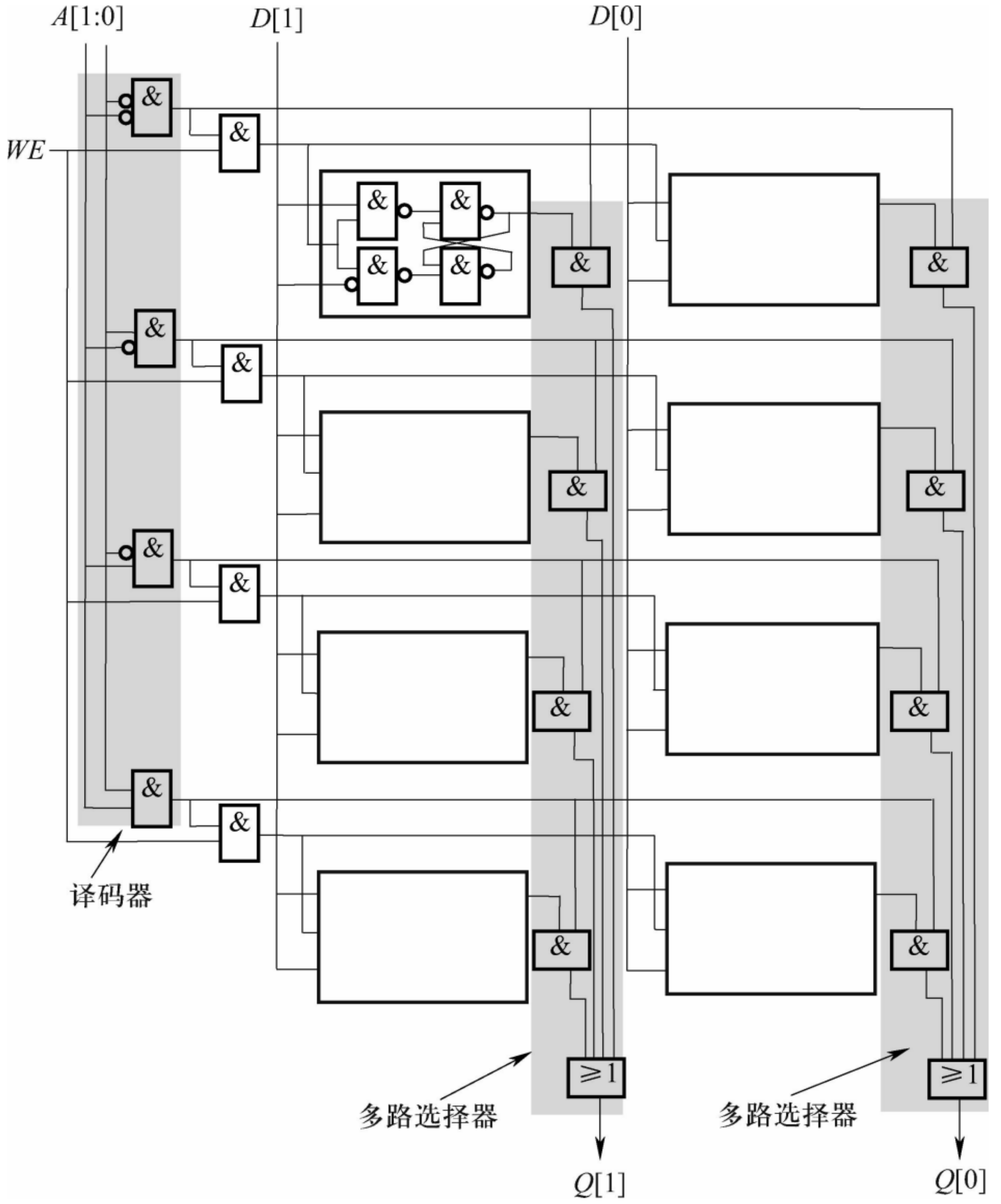


图 7.18 一个 4×2 位的存储器

南京大學
NANJING UNIVERSITY

03 运算方法和运算部件

1. 考虑如下C语言程序代码：

```c
int func1(unsigned word) {
  return (int)((word << 24) >> 24);
}


int func2(unsigned word) {
  return ((int)word << 24) >> 24;
}
```

假设在一个32位机器上执行这些函数，该机器使用二进制补码表示带符号整数。无符号数采用逻辑移位，带符号整数采用算数移位。

请填写下表缺失部分并说明函数func1和func2的功能。

W		**func1(w)**		**func2(w)**	
机器数	值	机器数	值	机器数	值
0000007FH	127	0000007FH	+127	0000007FH	+127
00000080H	128	00000080H	+128	**00000080H**	**-128**
000000FFH	255	000000FFH	+255	**000000FFH**	**-1**
00000100H	256	**00000100H**	**0**	**00000100H**	**0**

1. 考虑如下C语言程序代码：

```c
int func1(unsigned word) {
  return (int)((word << 24) >> 24);
}


int func2(unsigned word) {
  return ((int)word << 24) >> 24;
}
```

假设在一个32位机器上执行这些函数，该机器使用二进制补码表示带符号整数。无符号数采用逻辑移位，带符号整数采用算数移位。

请填写下表缺失部分并说明函数func1和func2的功能。

- 函数func1的功能是把无符号数高24位清零：逻辑左移 24位再逻辑右移24位，且结果一定是正的带符号整数
- 函数func2的功能是把无符号数的高24位都变成和第25位一样：因为左移24位后左边第一位变为原来的第25位，然后进行算术右移，高位补符号，即高24位都变成和原来第25位相同

2. 以下是两段C语言代码，函数arith()是直接用C语言写得，而optarith()是对函数arith()以某个确定的M和N编译生成的机器代码反编译生成的，根据optarith()，可以推断函数arith()中M和N的值各是多少?

```c
#define M
#define N


int arith(int x, int y) {
  int result = 0;
  result = x * M + y / N;
  return result;
}


int optarith(int x, int y) {
  int t = x;
  x << = 4;
  x - = t;
  if (y < 0) y + = 3;
  y >> = 2;
  return x + y;
}
```

⇨ x * M + y / N

⇨ x = x << 4 = x * 16

x = 16x - x = x * 15

y = y >> 2 = y / 4

⇨ M=15

N = 4

3. 设A4~A1和B4~B1分别是4位加法器的两组输入，C0为低位来的进位。
当加法器分别采用串行进位和先行进位时，写出4个进位C4、C3、C2、C1的逻辑表达式。

串行进位：

$$C_1 = A_1C_0 + B_1C_0 + A_1B_1$$

$$C_2 = A_2C_1 + B_2C_1 + A_2B_2$$

$$C_3 = A_3C_2 + B_3C_2 + A_3B_3$$

$$C_4 = A_4C_3 + B_4C_3 + A_4B_4$$

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

$G_i = A_iB_i$ 进位生成函数

$P_i = A_i \oplus B_i$ 进位传递函数

先行进位：

$$C_1 = A_1B_1 + (A_1 + B_1)C_0$$

$$C_2 = A_2B_2 + (A_2 + B_2)A_1B_1 + (A_2 + B_2)(A_1 + B_1)C_0$$

$$C_3 = A_3B_3 + (A_3 + B_3)A_2B_2 + (A_3 + B_3)(A_2 + B_2)A_1B_1 + (A_3 + B_3)(A_2 + B_2)(A_1 + B_1)C_0$$

$$C_4 = A_4B_4 + (A_4 + B_4)A_3B_3 + (A_4 + B_4)(A_3 + B_3)A_2B_2 + (A_4 + B_4)(A_3 + B_3)(A_2 + B_2)A_1B_1$$
$$+(A_4 + B_4)(A_3 + B_3)(A_2 + B_2)(A_1 + B_1)C_0$$

4. 请按照要求计算，并把结果还原成真值。

（1）设 $[x]_{补}$=0101、$[y]_{补}$=1101，求 $[x+y]_{补}$，$[x-y]_{补}$。

- $[x]_{补}$ = 0101，x 真值为5
- $[y]_{补}$ = 1101，y 真值为-3

➢ $[x+y]_{补}$ = $[x]_{补}+[y]_{补}$= (1)0010=2

➢ $[x-y]_{补}$ = $[x]_{补}+[-y]_{补}$ = $[x]_{补}+\left[[y]_{补}\right]_{补'}$

= 0101+0011=1000=-8

➢ $x+y=2$，$[x+y]_{补}=0010$

➢ $x-y=8$(溢出)，$[x-y]_{补}=1000$

4. 请按照要求计算，并把结果还原成真值。

（2）设 $[x]_{原}$=0101、 $[y]_{原}$=1101，用原码一位乘法计算 $[x*y]_{原}$。

原码乘法的运算方法

- $|x|_{原}$=0101
- $|y|_{原}$=1101

- 符号单独运算：直接异或
- 绝对值相乘：仅需考虑数值部分的计算
- 逐位相乘，错位相加
  - 先计算相加数，然后按列求和

					0	1	0	1
				0	0	0	0	
			0	1	0	1		
*		0	0	0	0			
=	1	0	0	1	1	0	0	1

符号位为1， 因此 $[x*y]_{原}$ = 10011001，真值为-25

4. 请按照要求计算，并把结果还原成真值。

（3）设 $[x]_{补}$=0101， $[y]_{补}$=1101，用补码一位布斯乘法计算 $[x*y]_{补}$。

(布斯乘) 假设X,Y为被乘数和乘数，x, y为它们的位数，中间结果A、S和乘法结果P都是(x+y+1)位的长度

I. 计算中间结果 A (short for addition) 和 S (short for subtraction) 并初始化 P (short for production)：
   1. A　被乘数X放在高x位上，低y+1位补0；
   2. S　-X的补码 $[-x]_{补}$ 放在高x位上，低y+1位补0；
   3. P　在高x位上补0，接着的y位放乘数Y，最后一位补0；

II. 根据P的最低两位进行运算：
   1. 如果是01，计算 P=P+A 的值；
   2. 如果是10，计算 P=P+S 的值；
   3. 如果是00，不做运算，P=P；
   4. 如果是11，不做运算，P=P；

III. 将上一步得到的 P 算术右移1位，P = P >> 1

IV. 重复执行步骤II和III，共进行 y 次后停止

V. 最后，截断 P 的最低一位，得到 X 和 Y 的乘积

$[-x]_{补} = 1011$

```
A = 0101 0000 0
S = 1011 0000 0
P = 0000 1101 0
```

```
① P=P+S, P>>1, P = 1101 1110 1
② P=P+A, P>>1, P = 0001 0111 0
③ P=P+S, P>>1, P = 1110 0011 1
④ P=P,   P>>1, P = 1111 0001 1
```

可得 $[x*y]_{补}$ = 11110001，真值为-15

4. 请按照要求计算，并把结果还原成真值。

（3）设 $[x]_{补}$=0101， $[y]_{补}$=1101，用补码一位布斯乘法计算 $[x*y]_{补}$。

- 符号位参与运算
- Y_0Y_{-1}= 00 或 11： $\sum += 0$
- Y_0Y_{-1}= 01： $\sum += [x]_{补}$
- Y_0Y_{-1}= 10： $\sum += [-x]_{补}$

因此 $[x*y]_{补}$ = 11110001，真值为-15