

Arithmetic  
&  
Logical Op

Binary Adder, Binary Subtractor, Binary Multiplier

Binary Comparator (Magnitude Comparator)

Data  
Transmission

Decoder, Encoder

**Multiplexer (MUX, MPX), De-Multiplexer (Demux)**

Coders

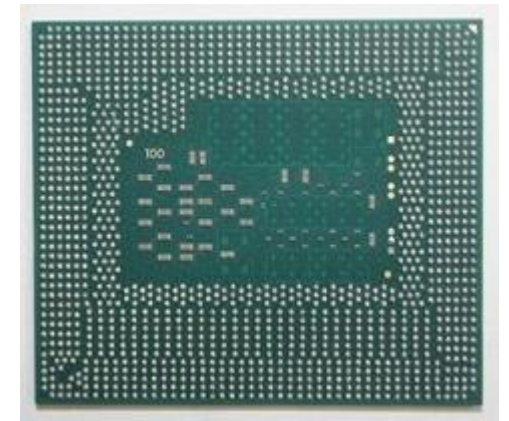
Binary Codes (BCD, Excess-3, Gray)

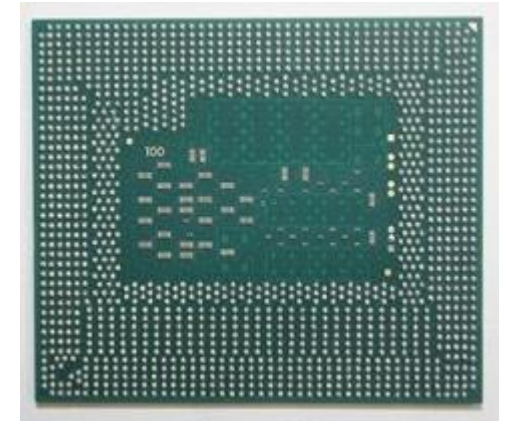
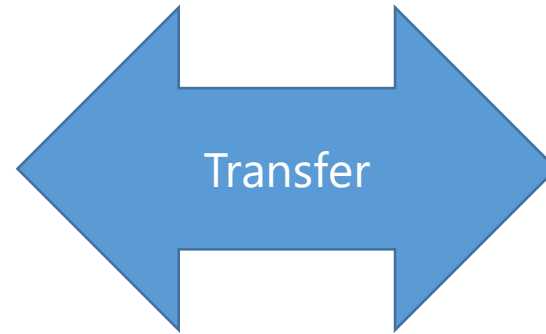
---

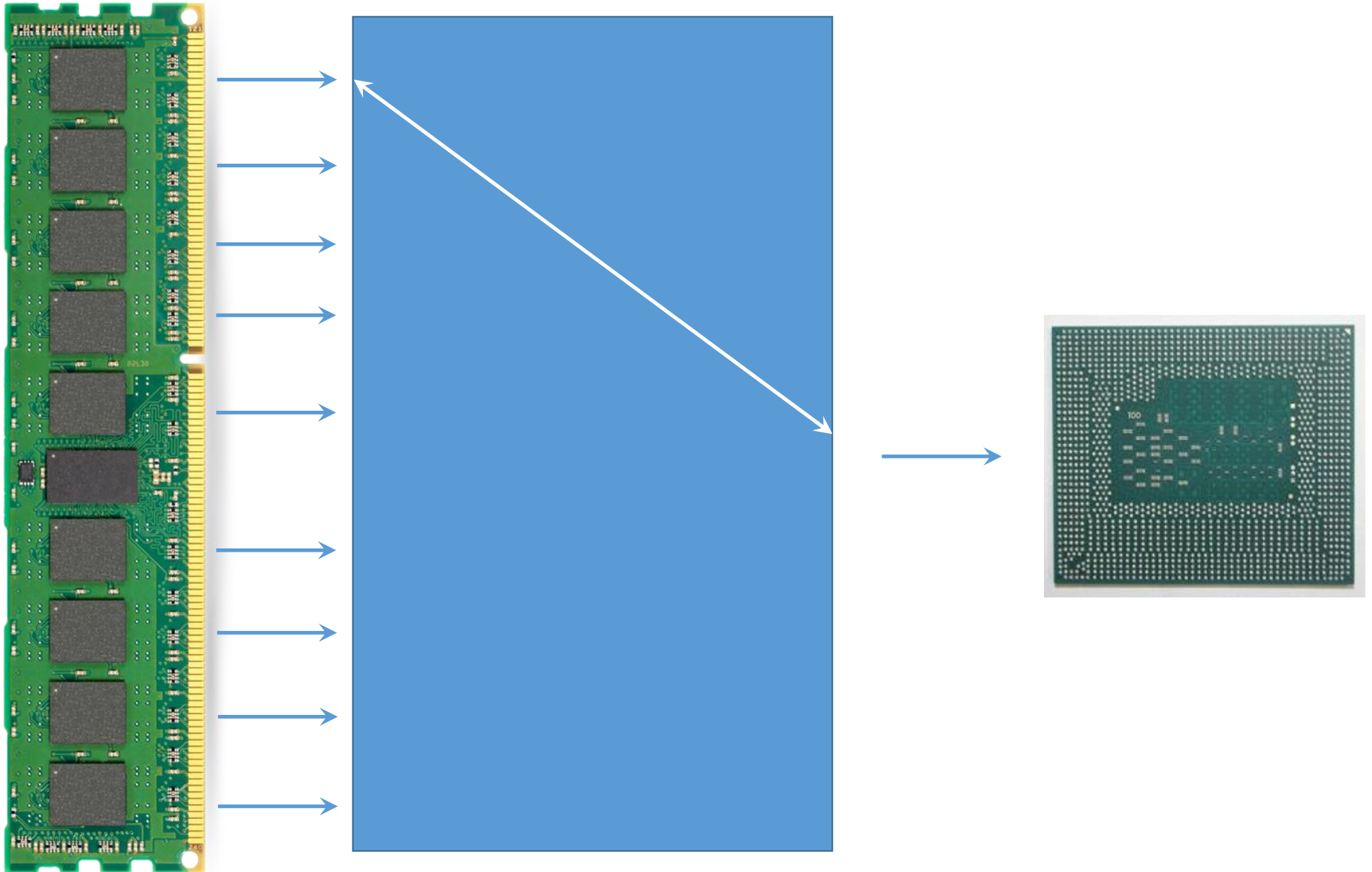
# Multiplexer

Shortened to MUX or MPX

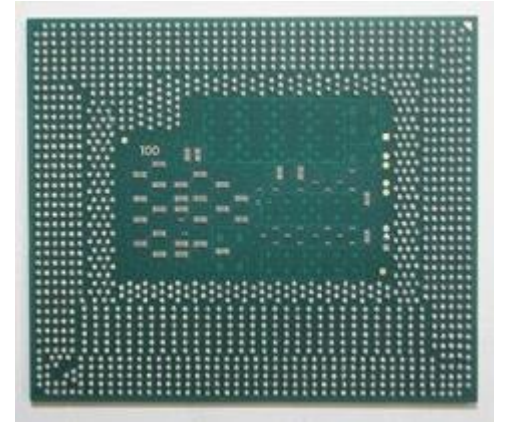
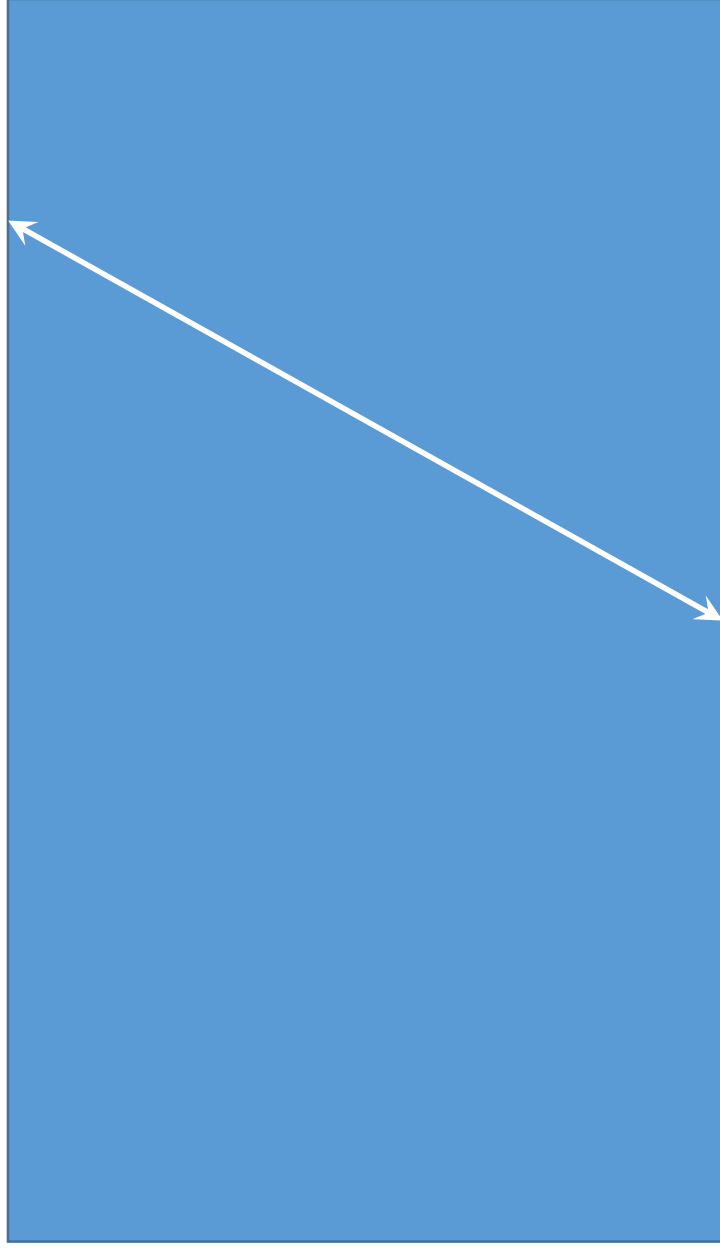
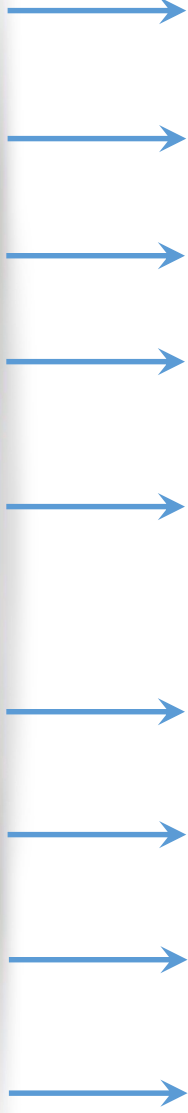
---

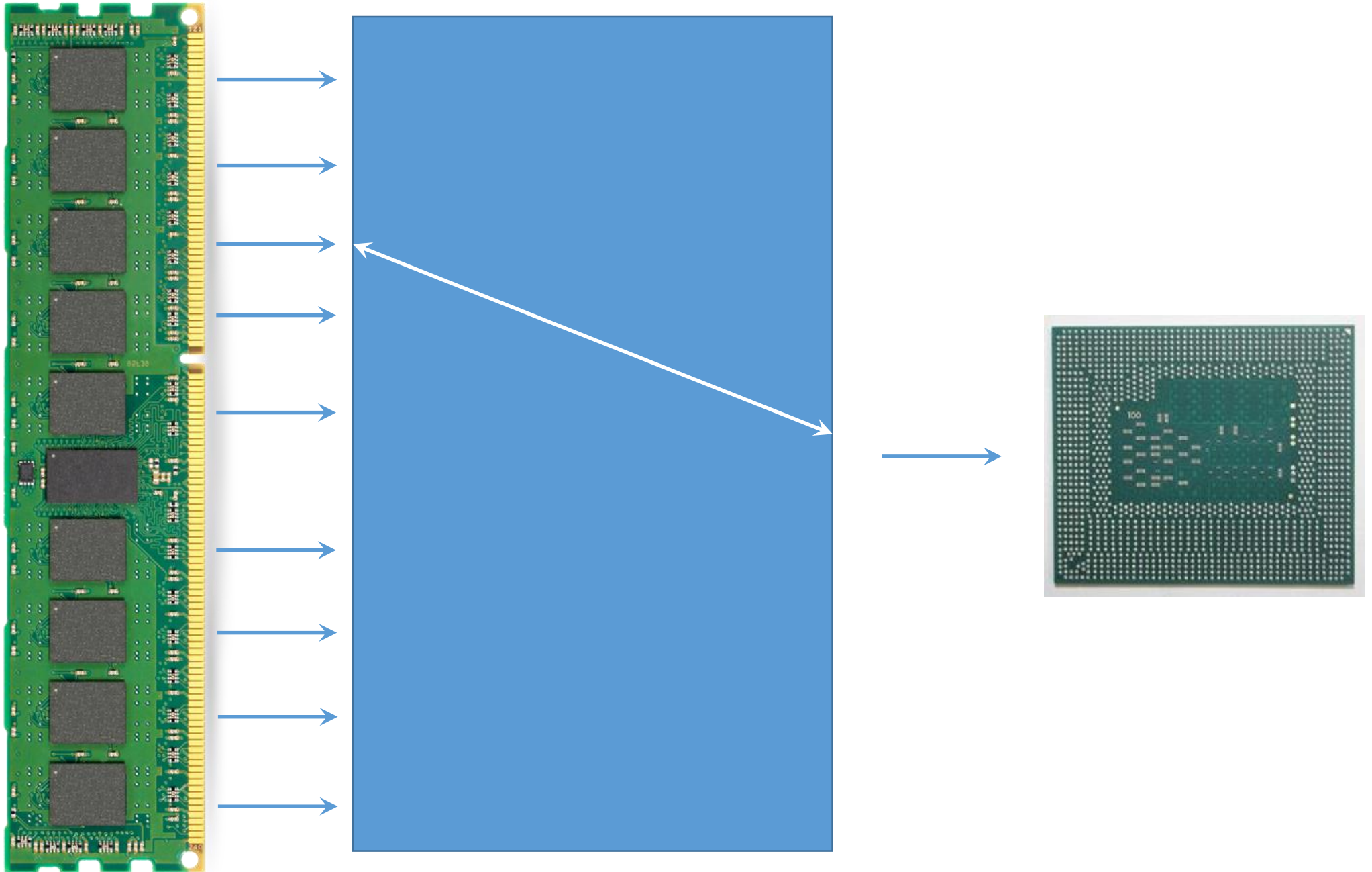


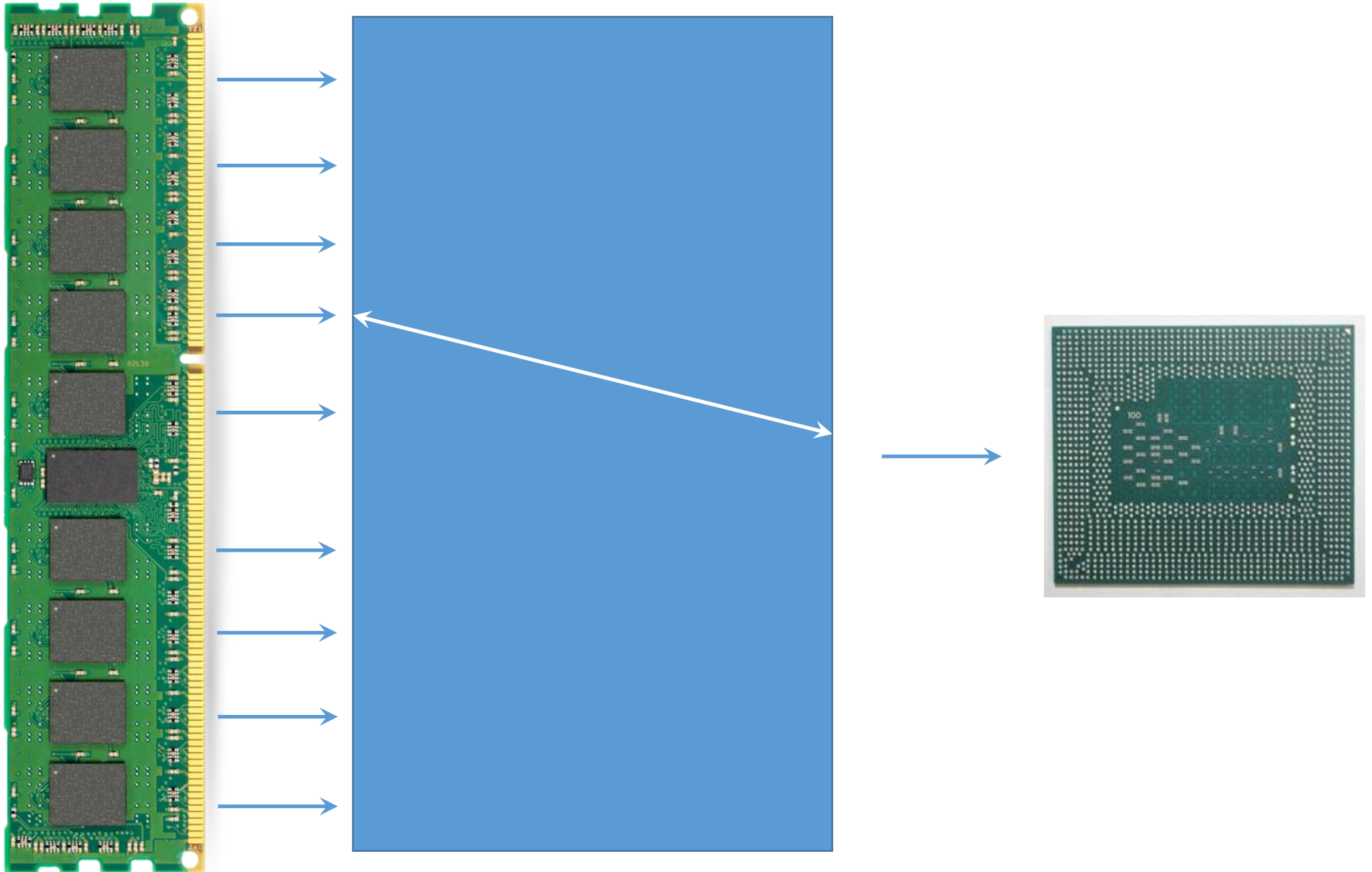




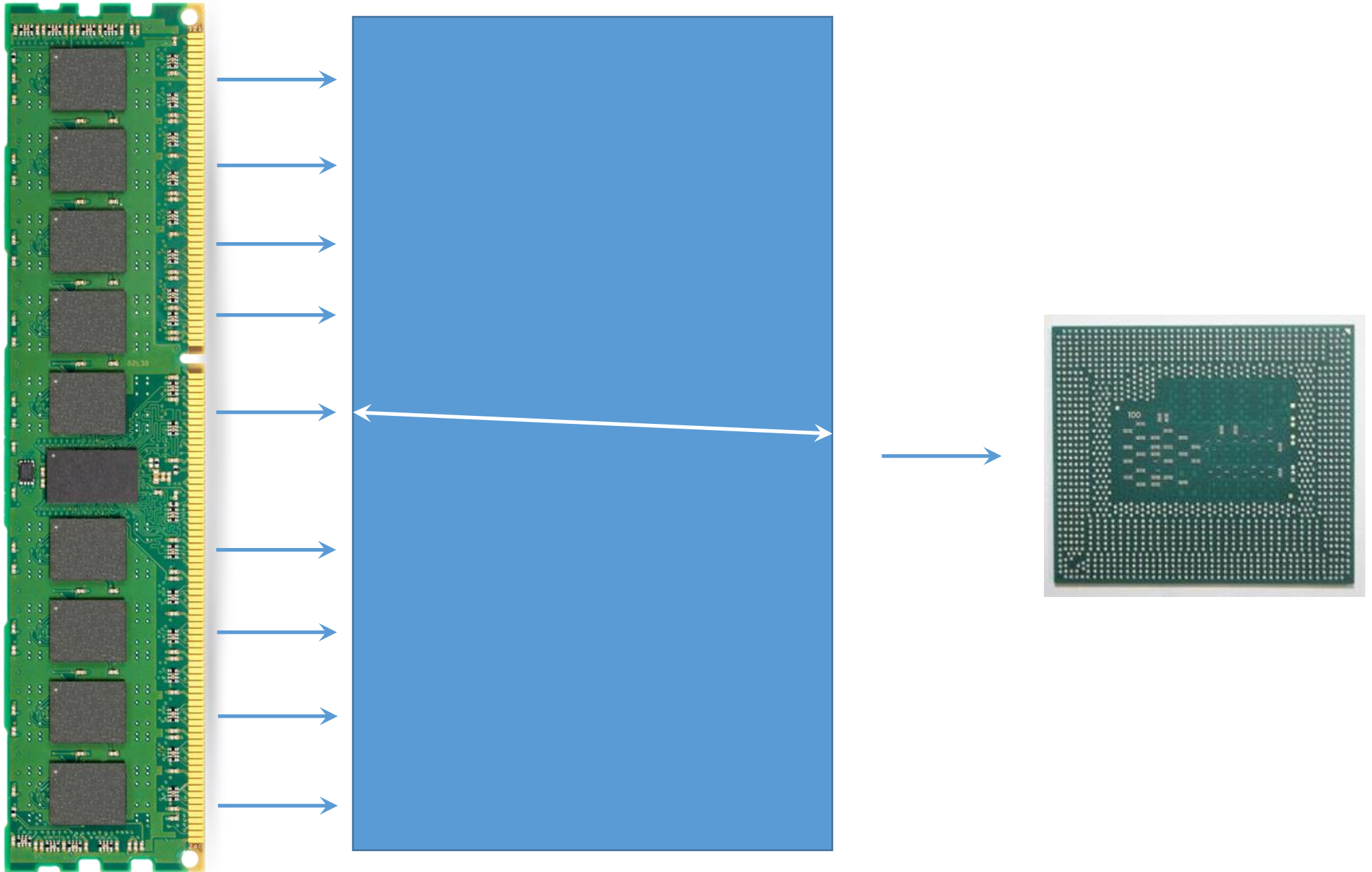


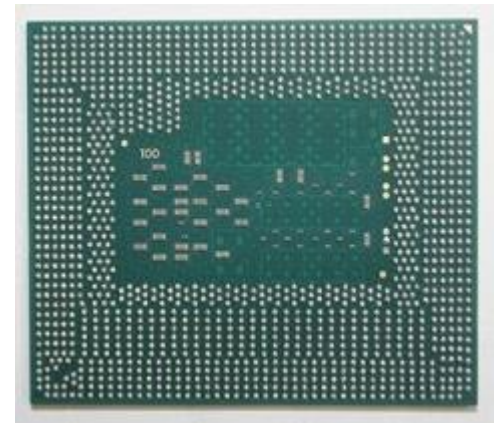
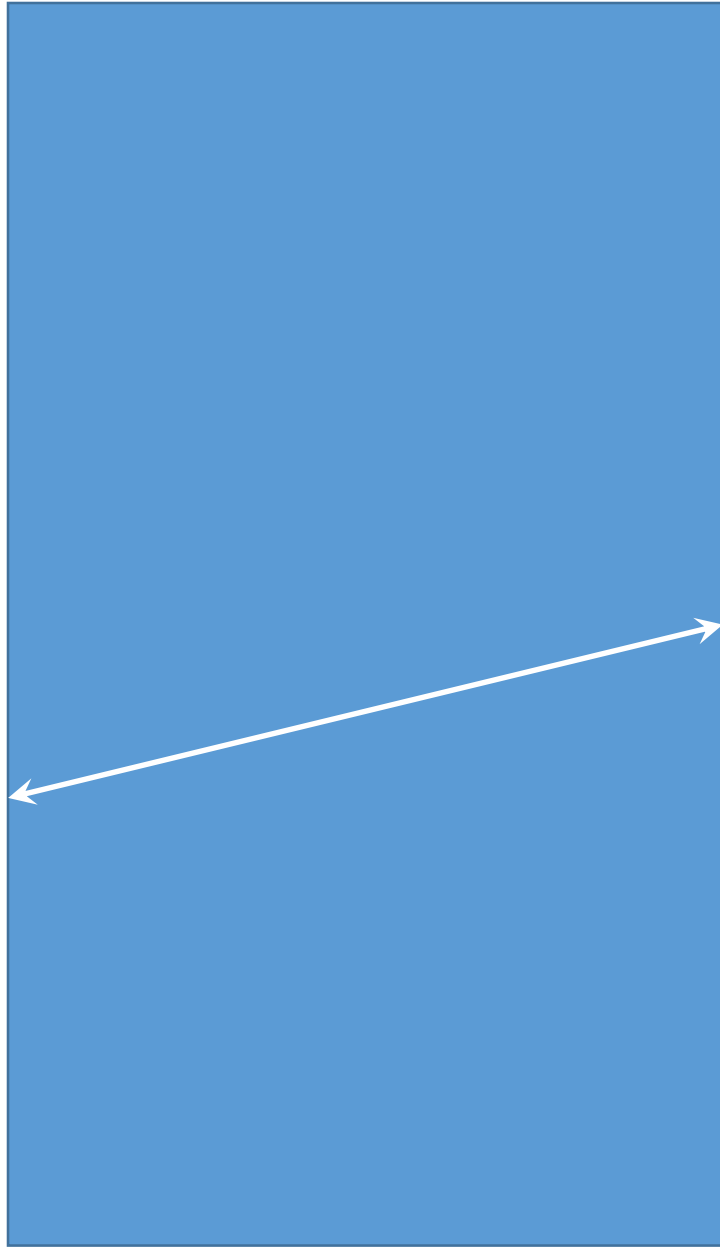
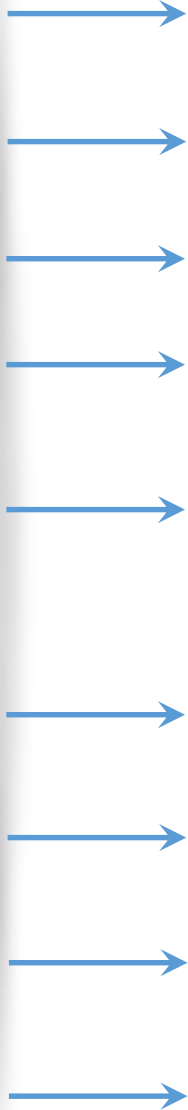


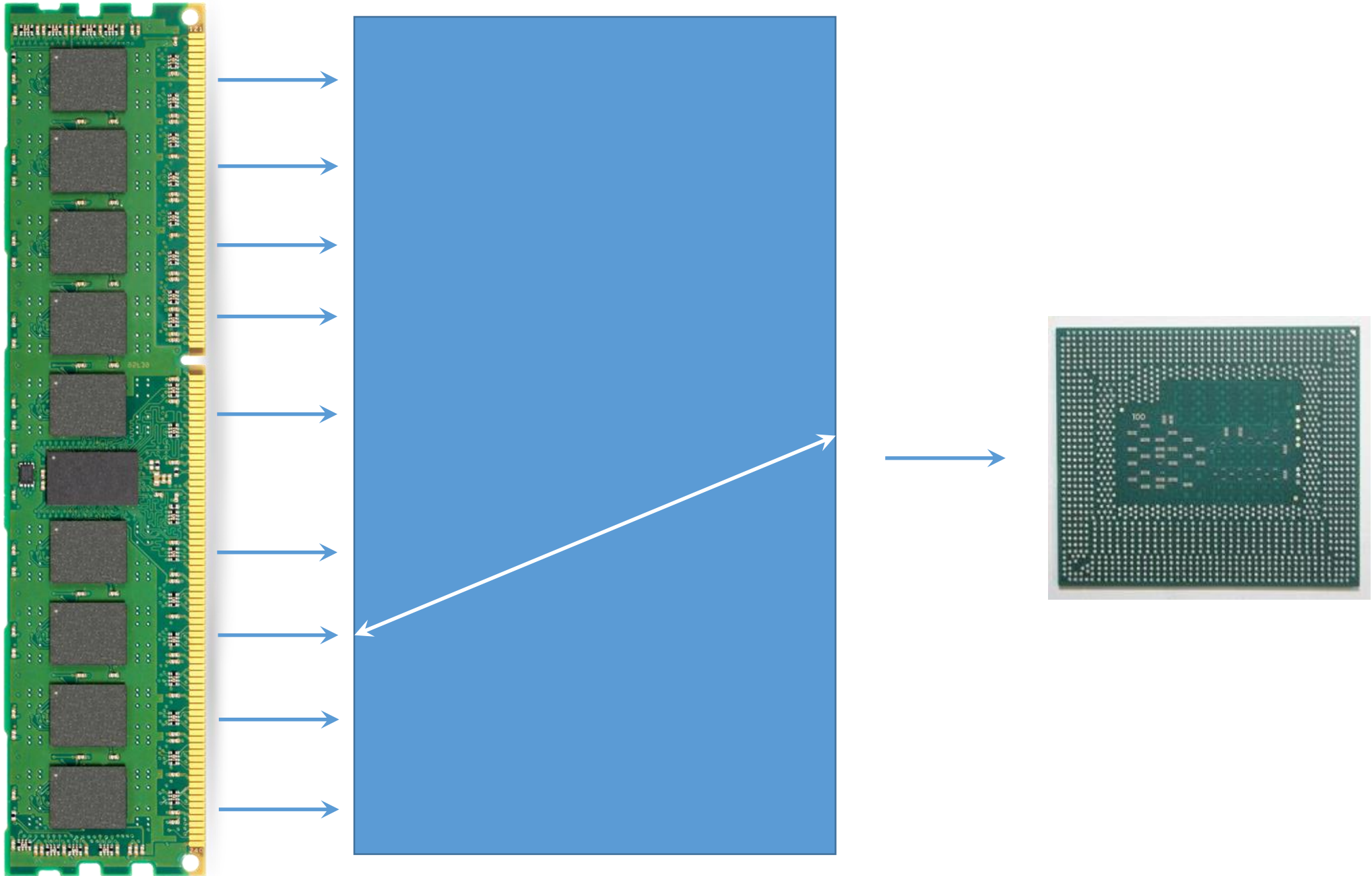


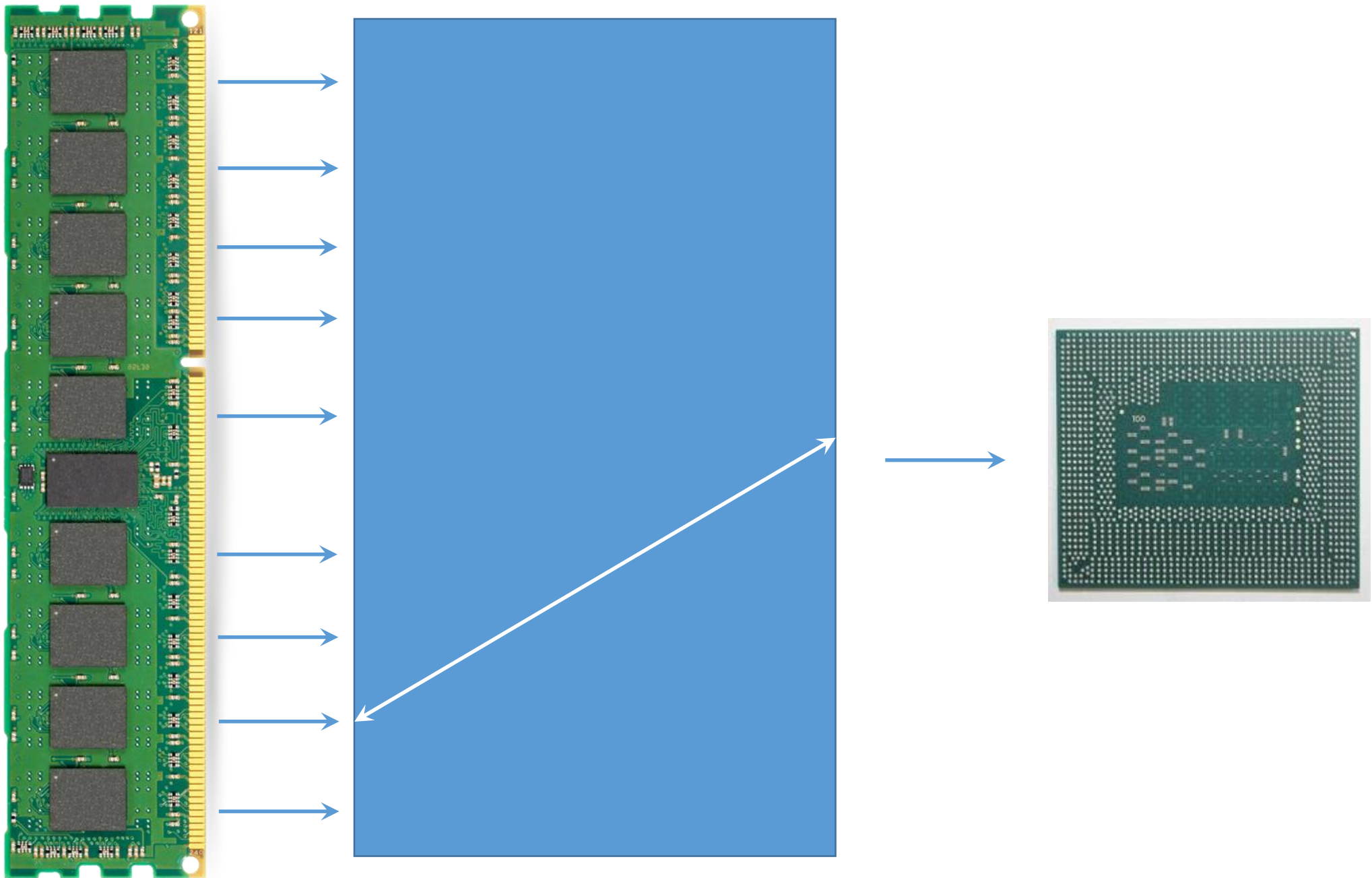




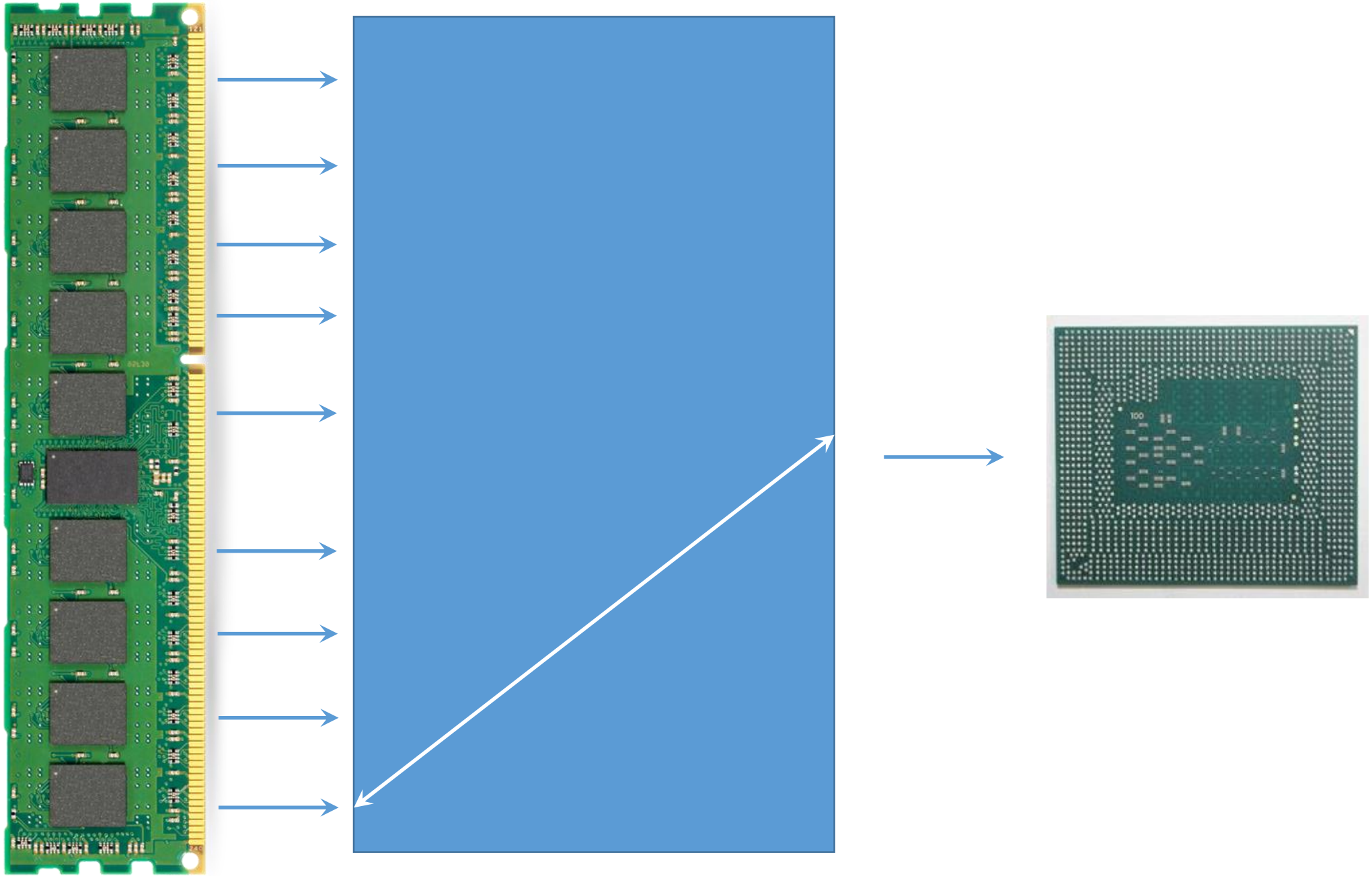












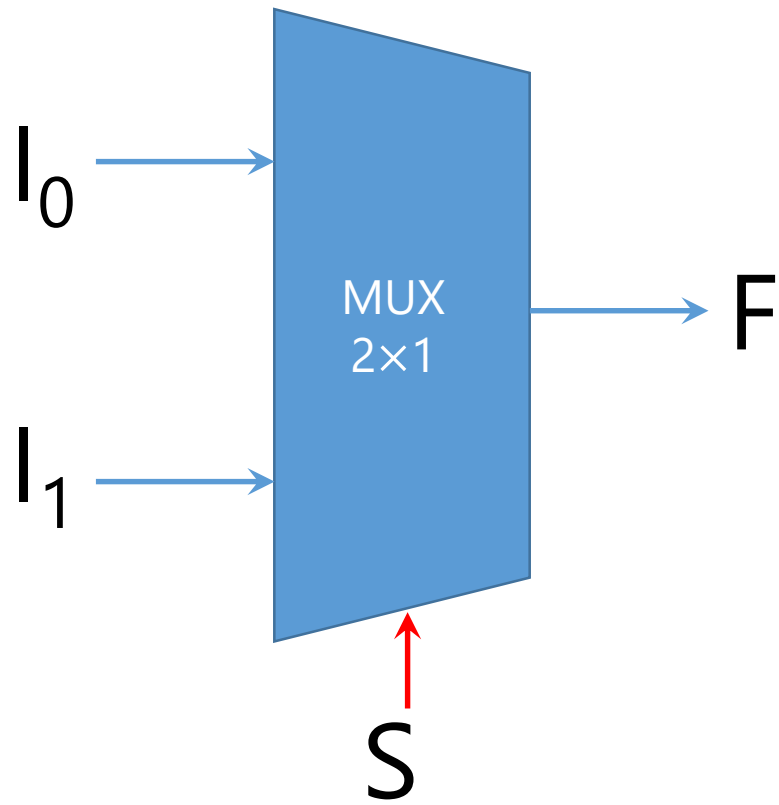


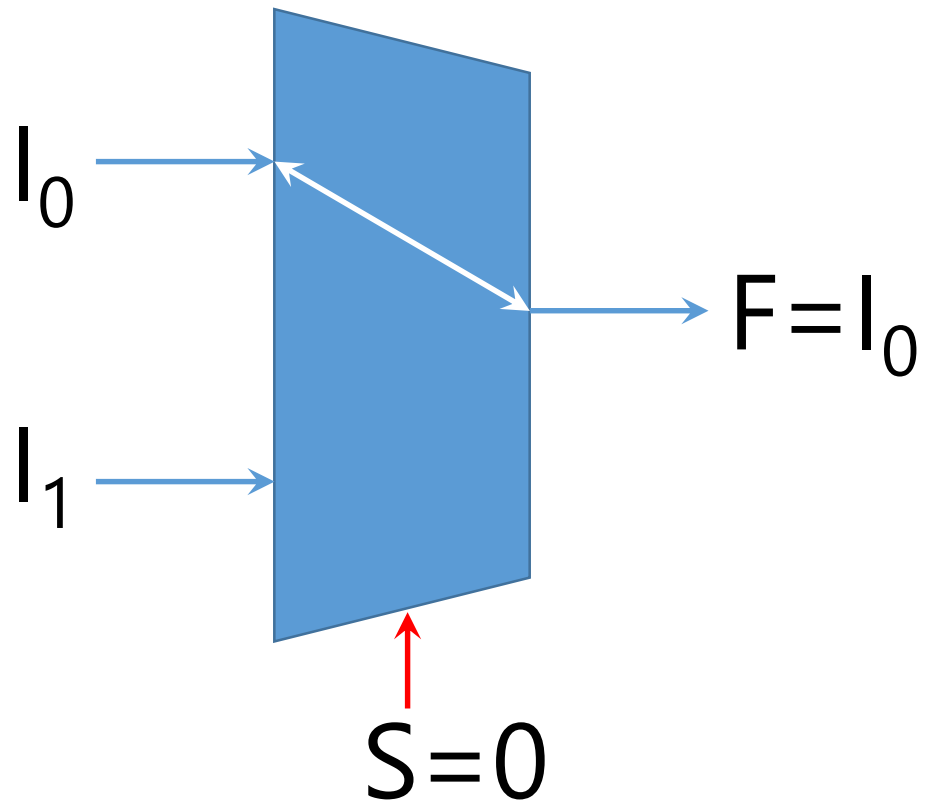
---

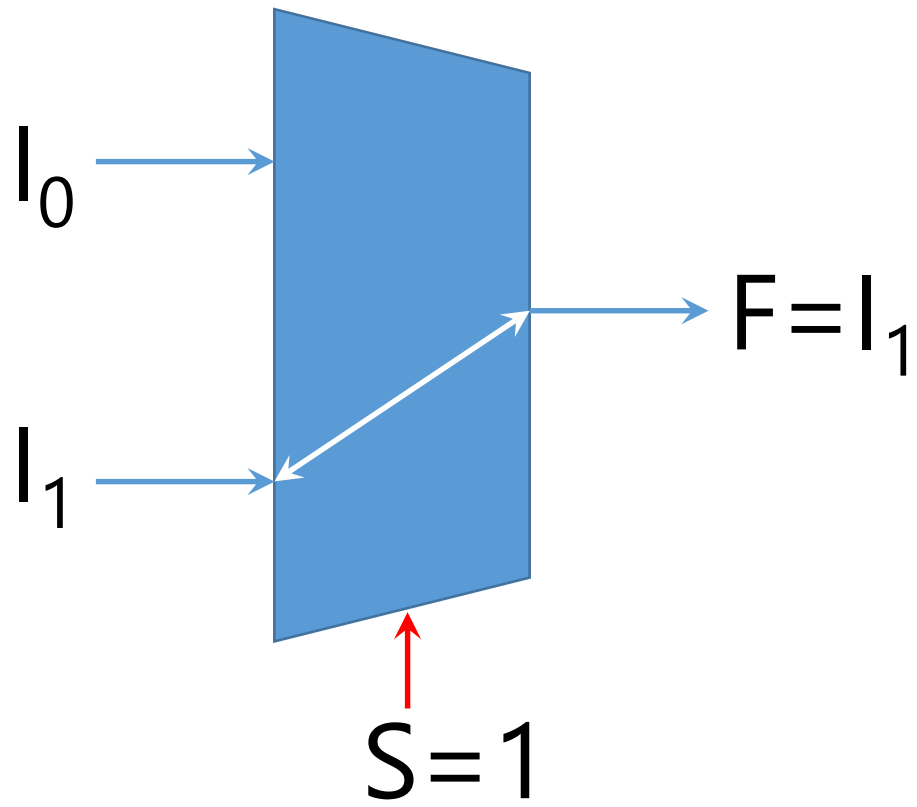
# Multiplexer

## $2^1 \times 1$

---







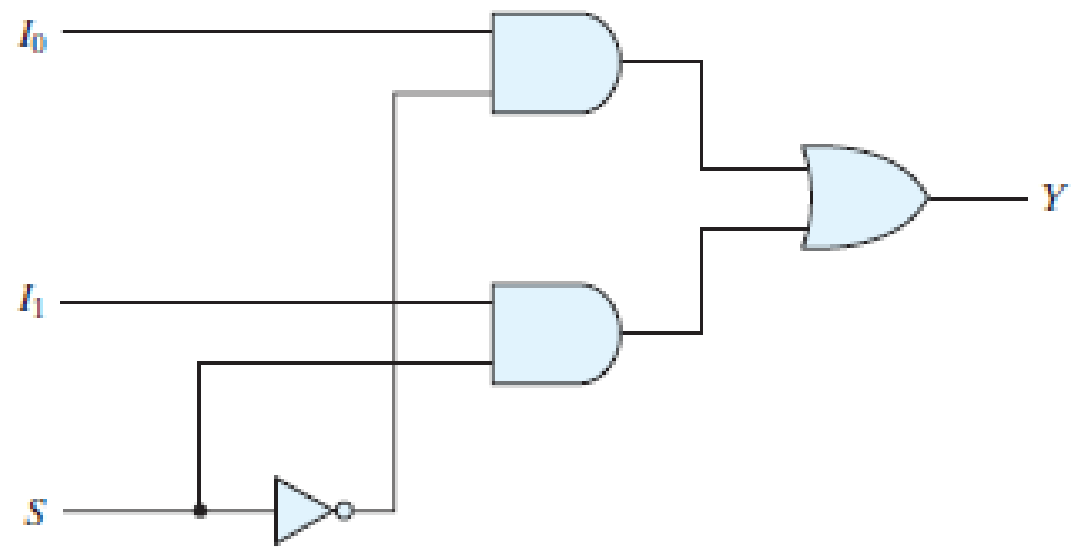
S	I <sub>1</sub>	I <sub>0</sub>	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



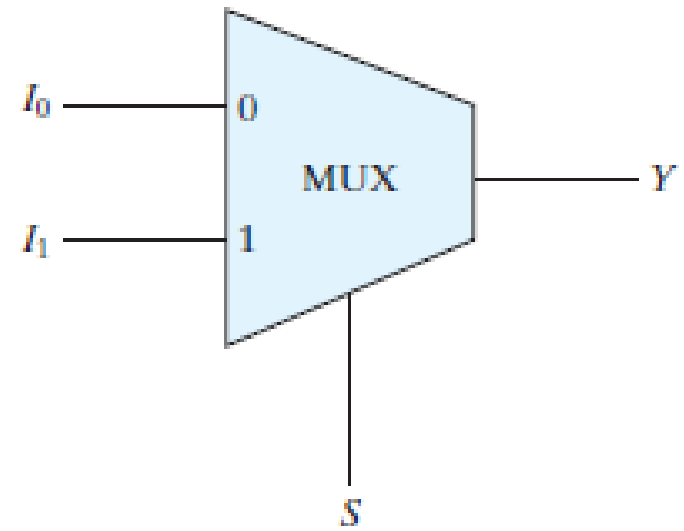
S	I <sub>1</sub>	I <sub>0</sub>	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

		$I_1 I_0$			
		00	01	11	10
S	0	0 $m_0$	1 $m_1$	1 $m_3$	0 $m_2$
	1	0 $m_4$	0 $m_5$	1 $m_7$	1 $m_6$

$$F = S'I_0 + SI_1$$



(a) Logic diagram



(b) Block diagram

**FIGURE 4.24**  
Two-to-one-line multiplexer

S	$F = S'I_0 + SI_1$
0	$I_0$
1	$I_1$

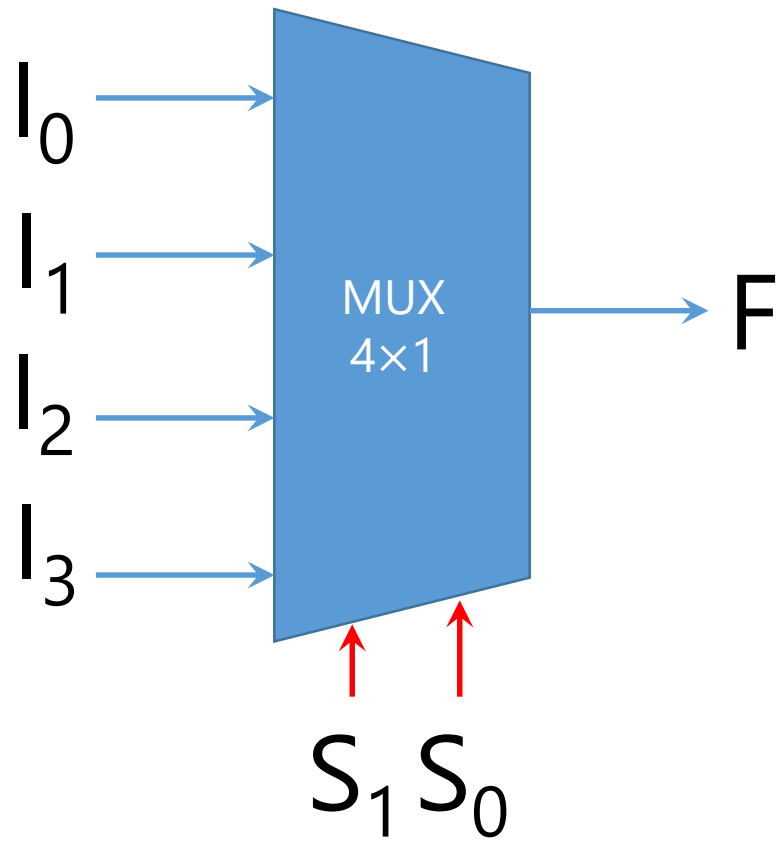
---

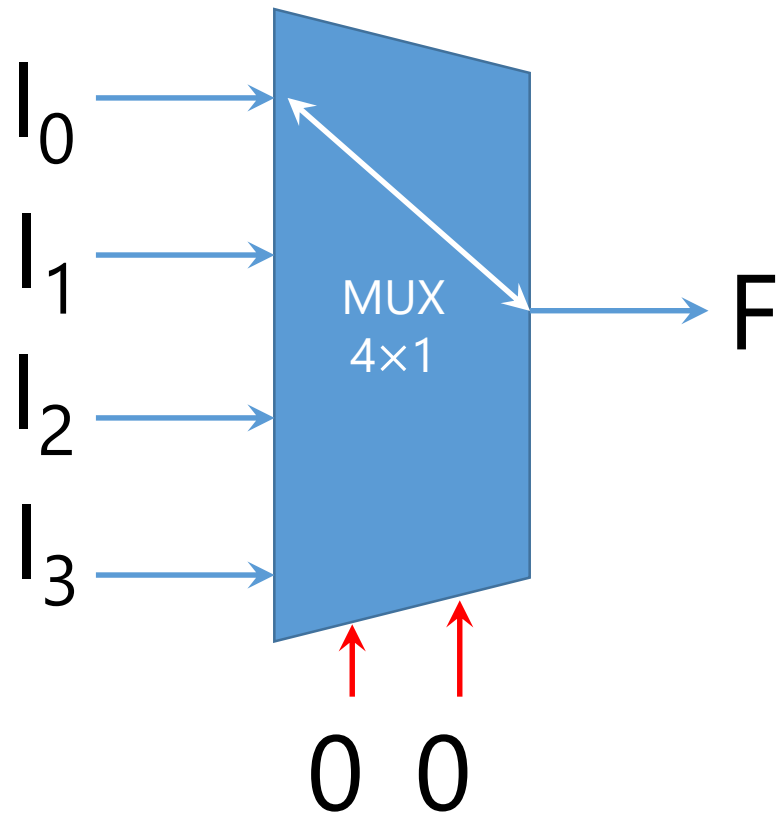
# Multiplexer

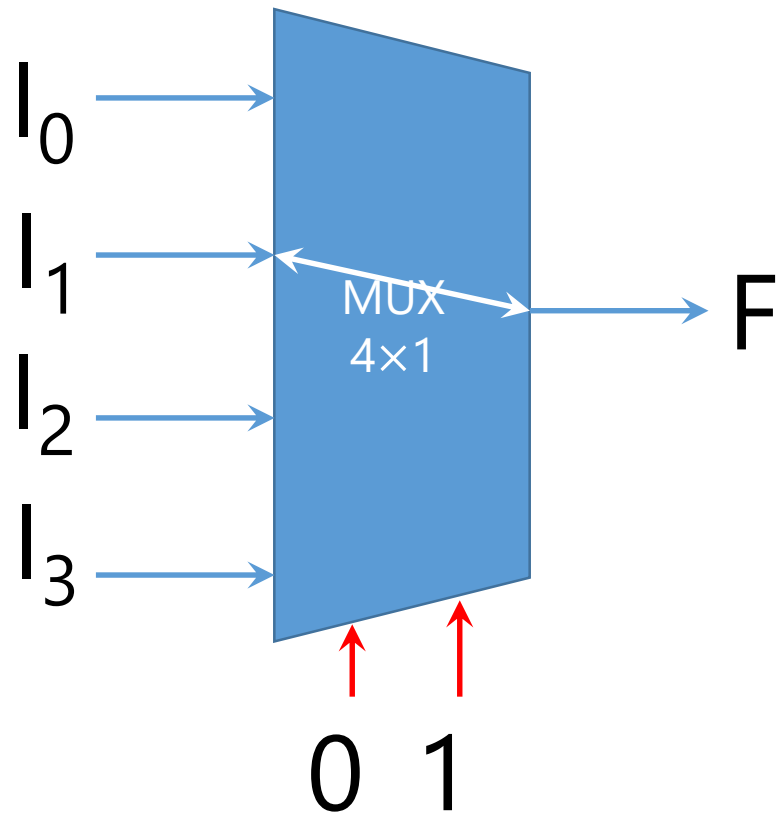
## $2^2 \times 1$

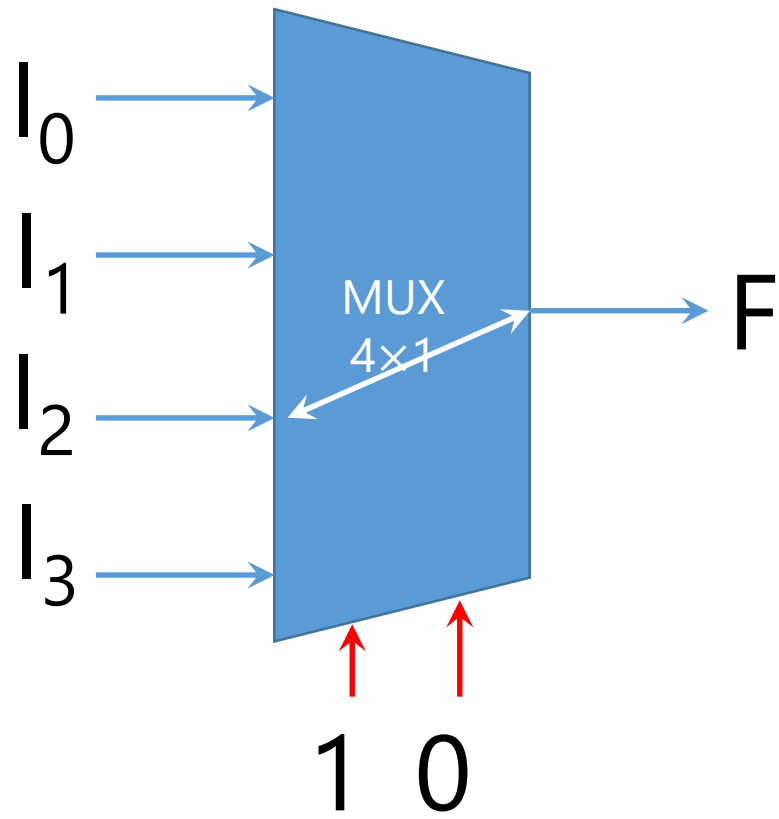
---

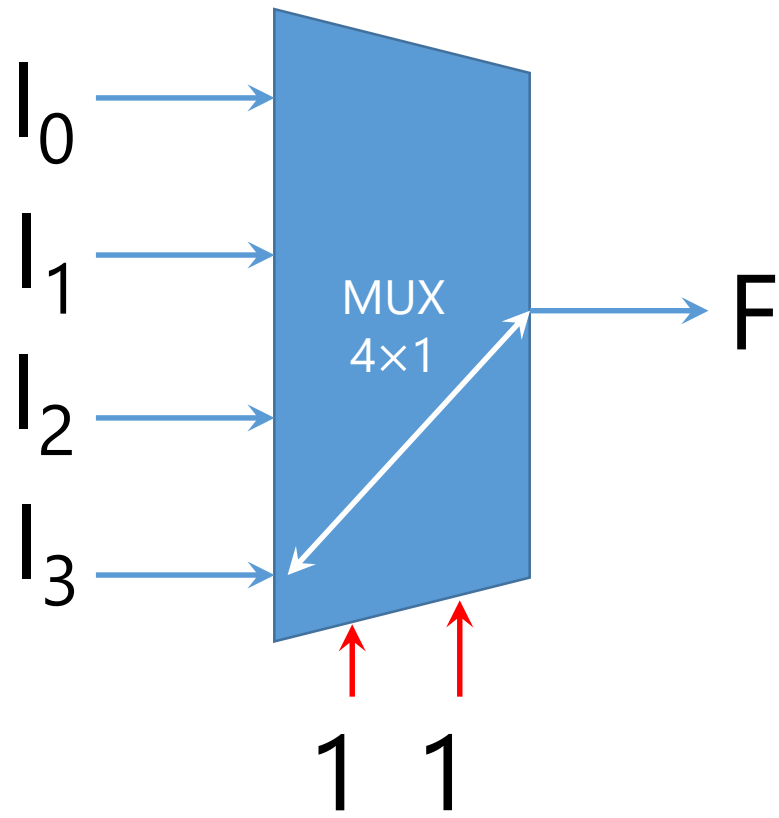








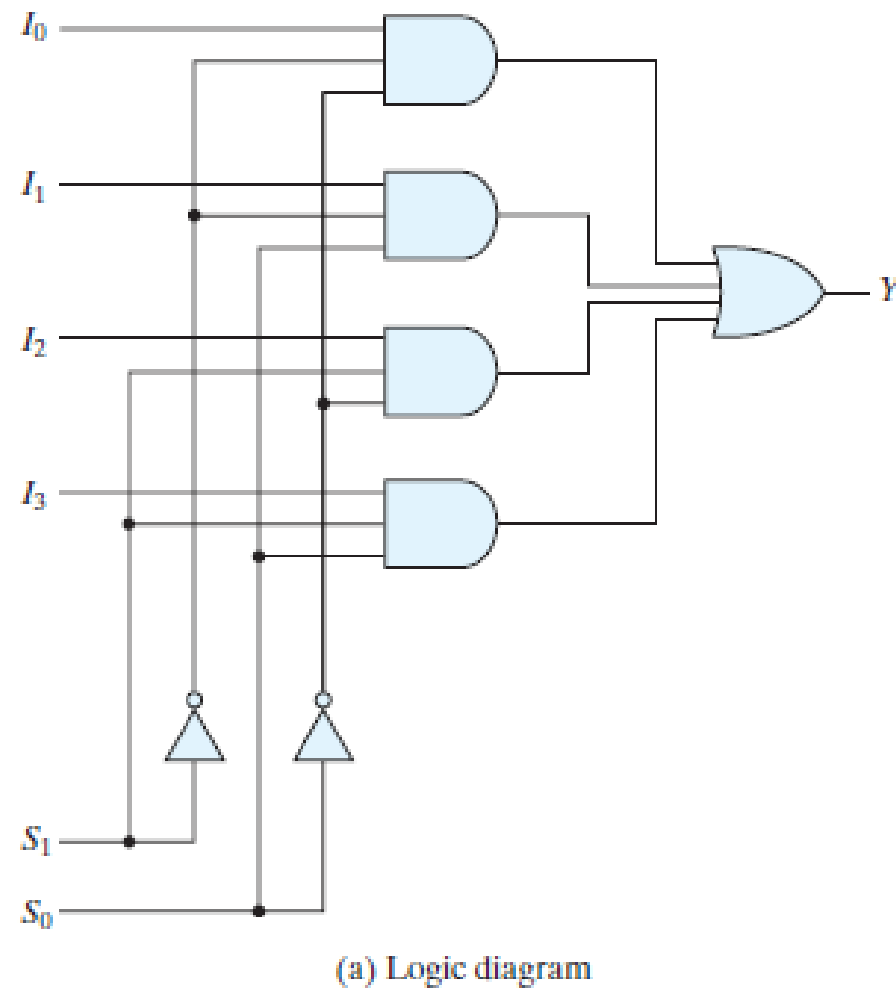






$S_1$	$S_0$	$I_3$	$I_2$	$I_1$	$I_0$	F
0	0	x	x	x	0	0
0	0	x	x	x	1	1
0	1	x	x	0	x	0
0	1	x	x	1	x	1
1	0	x	0	x	x	0
1	0	x	1	x	x	1
1	1	0	x	x	x	0
1	1	1	x	x	x	1

$S_1$	$S_0$	$F = S'_1 S'_0 I_0 + S'_1 S_0 I_1 + S_1 S'_0 I_2 + S_1 S_0 I_3$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$



$S_1$	$S_0$	$Y$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$

(b) Function table

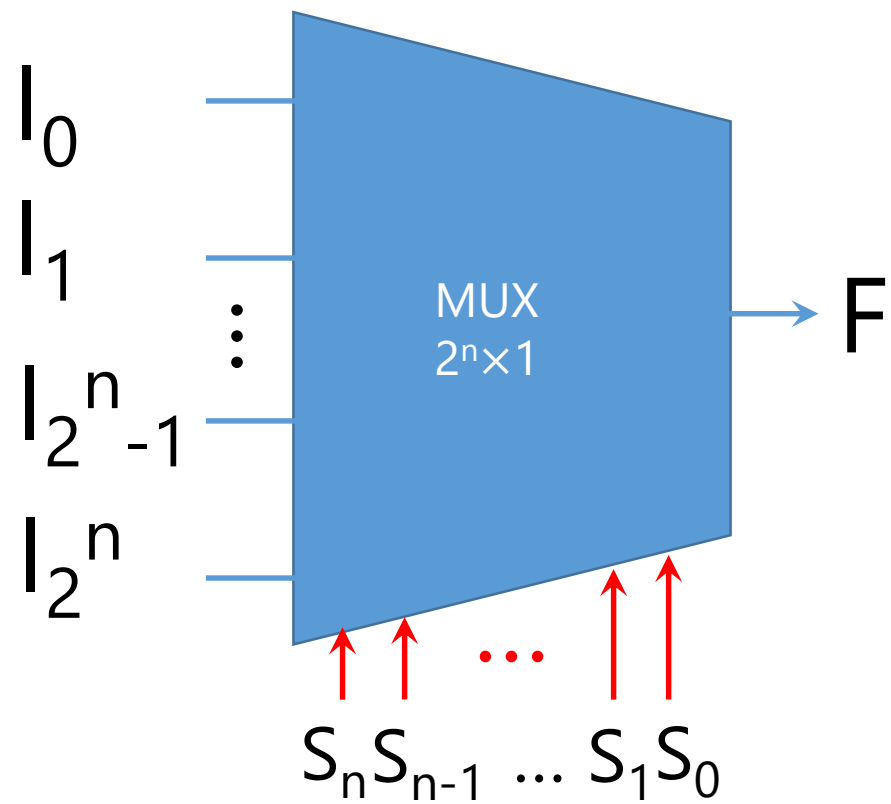
**FIGURE 4.25**  
Four-to-one-line multiplexer

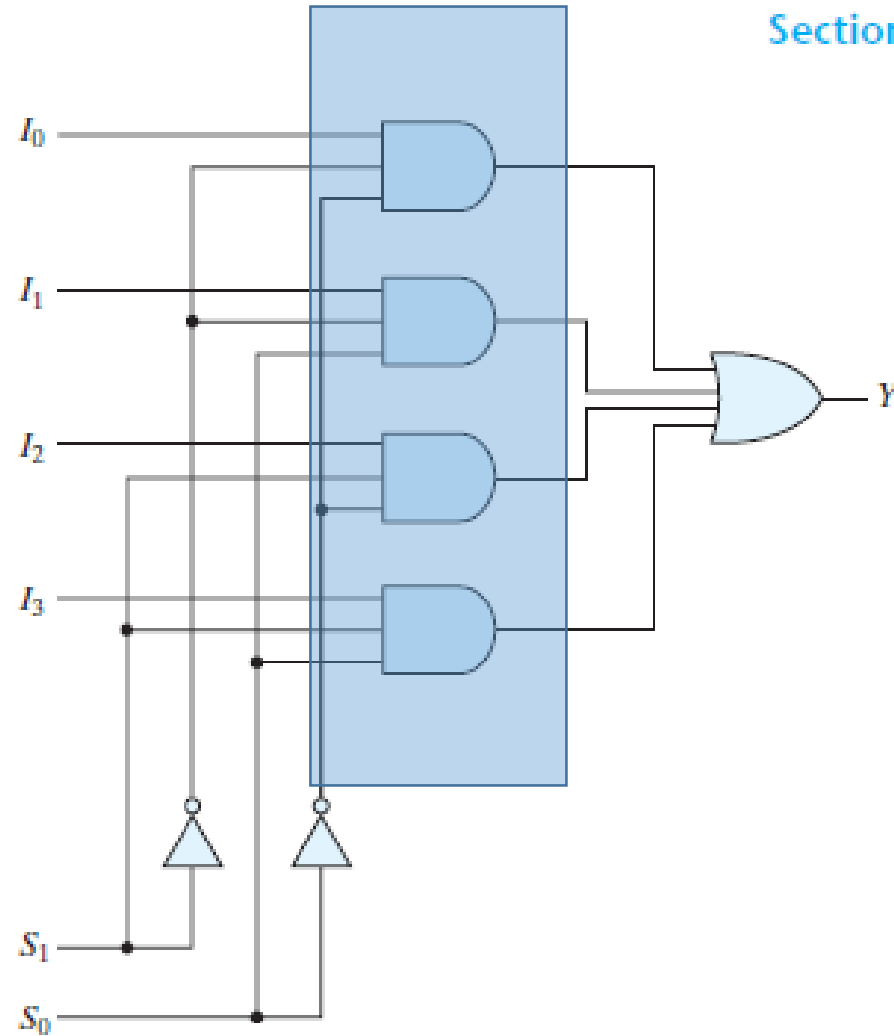
---

# Multiplexer

## $2^n \times 1$

---





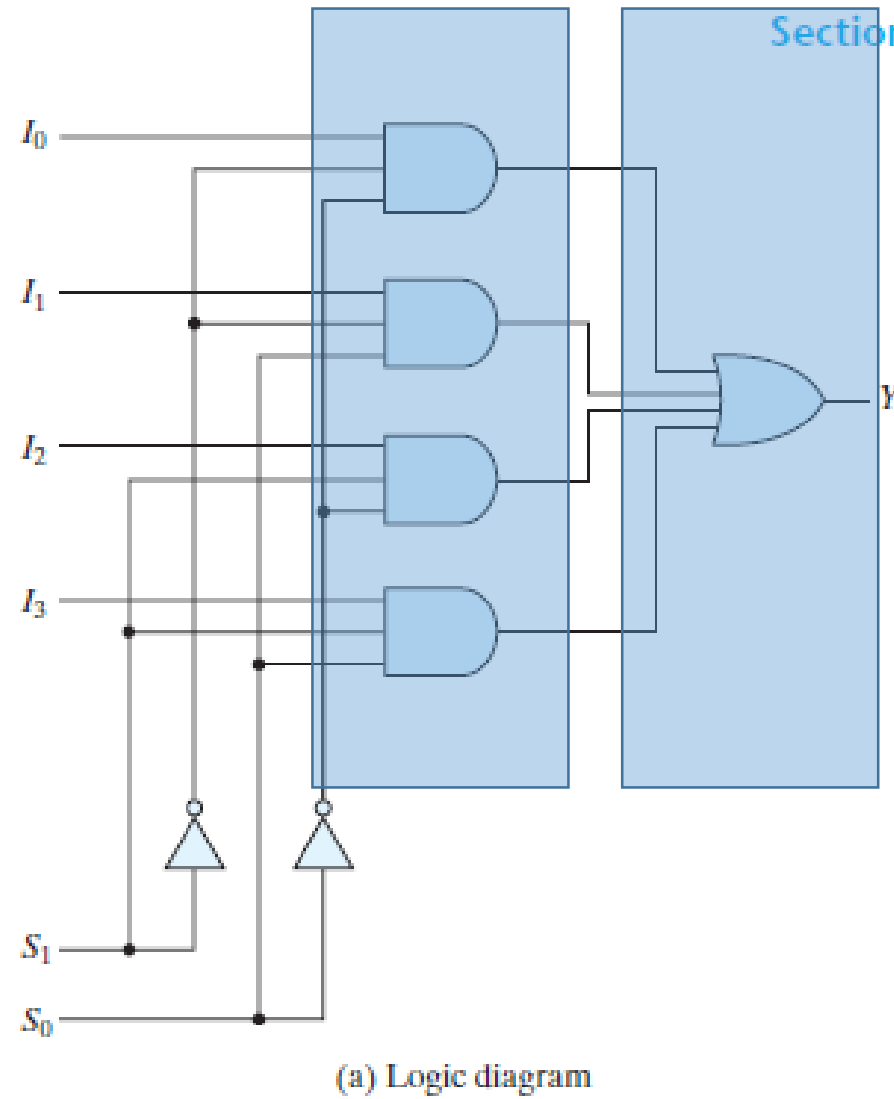
(a) Logic diagram

$\approx$  Decoder + OR

$S_1$	$S_0$	$Y$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$

(b) Function table

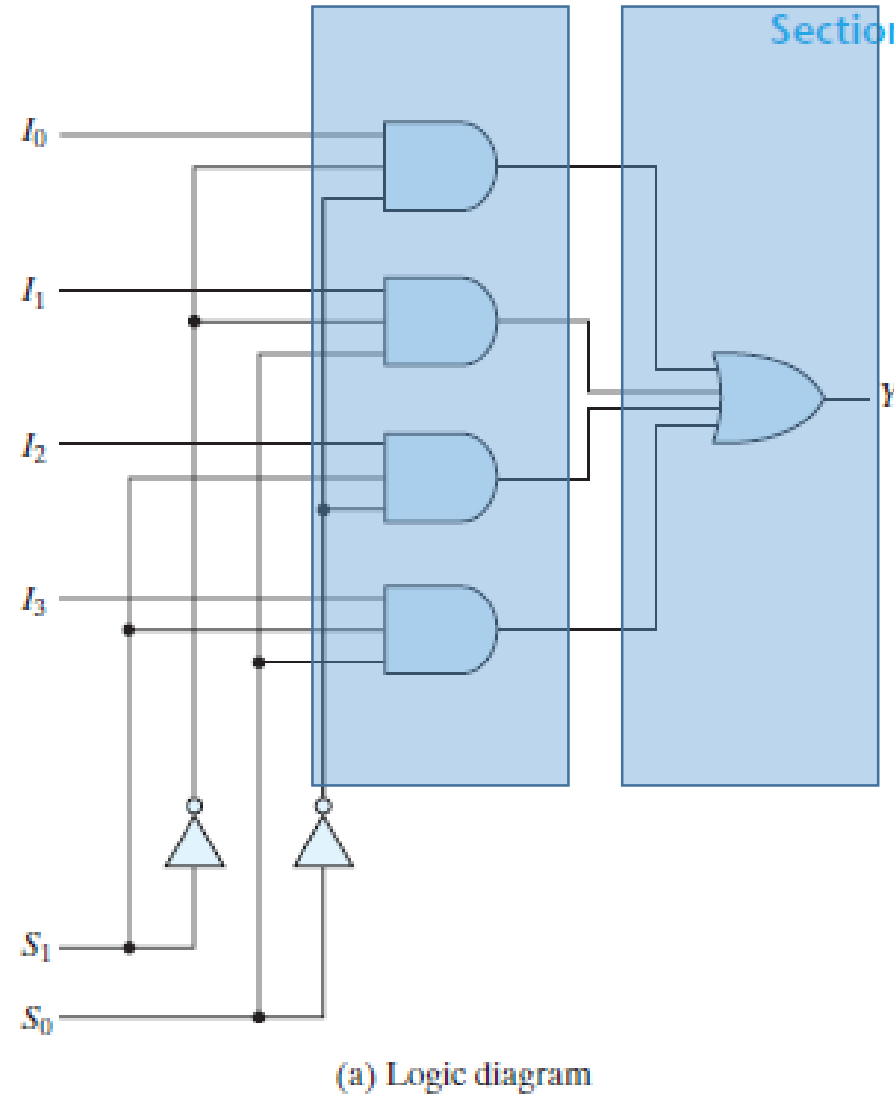
**FIGURE 4.25**  
Four-to-one-line multiplexer



$S_1$	$S_0$	$Y$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$

(b) Function table

**FIGURE 4.25**  
Four-to-one-line multiplexer



Sum of Products  
2 Levels  
ANDs-OR

$S_1$	$S_0$	$Y$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$

(b) Function table

**FIGURE 4.25**  
Four-to-one-line multiplexer



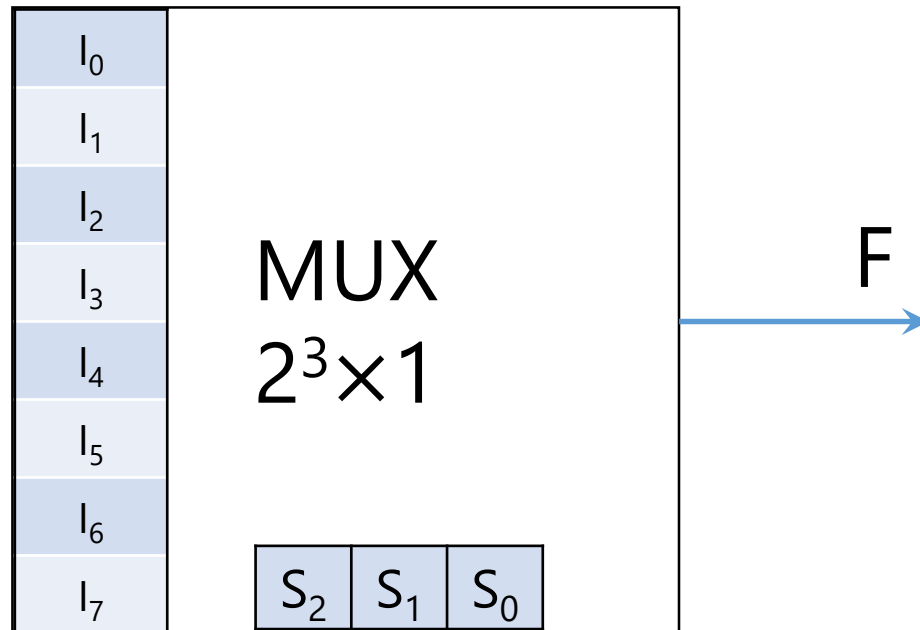
---

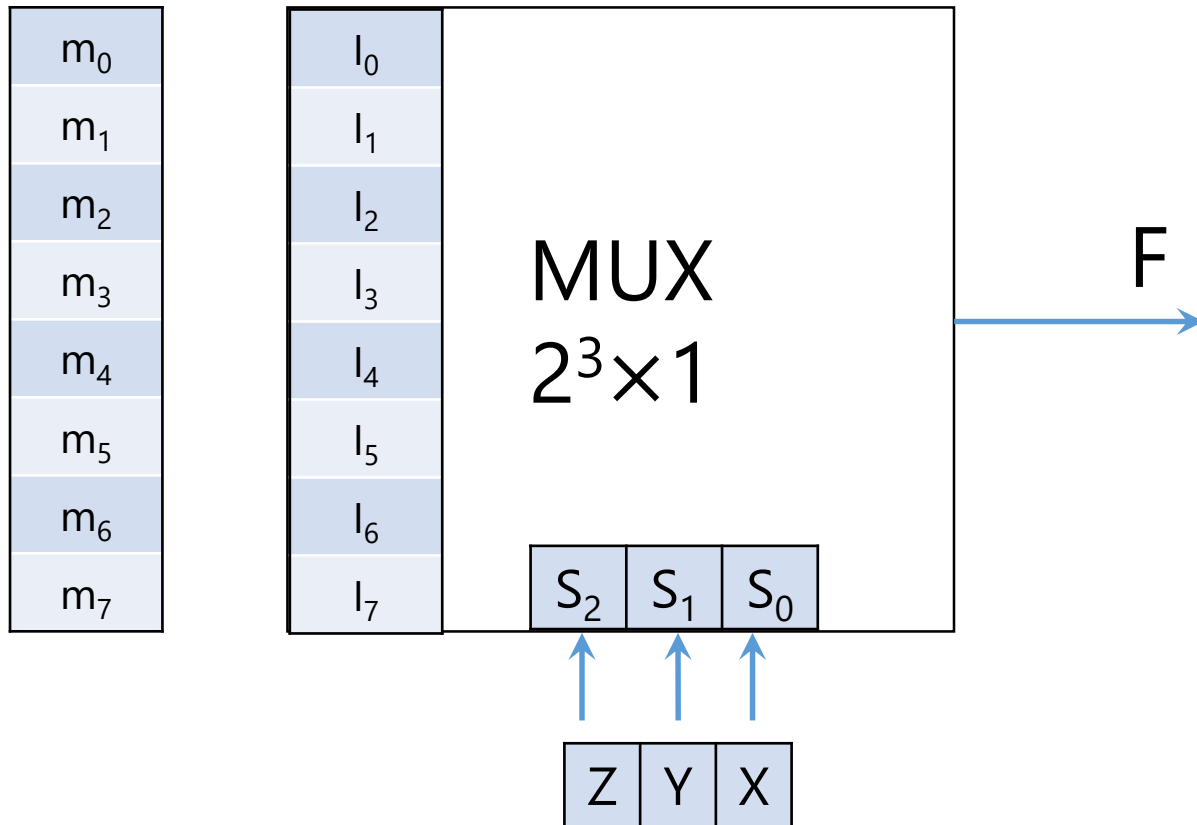
# Multiplexer Boolean Function

---

$$F_{\text{SoP}} = \sum m(\dots)$$

$$F_{\text{PoS}} = \prod M(\dots)$$





---

# MUX

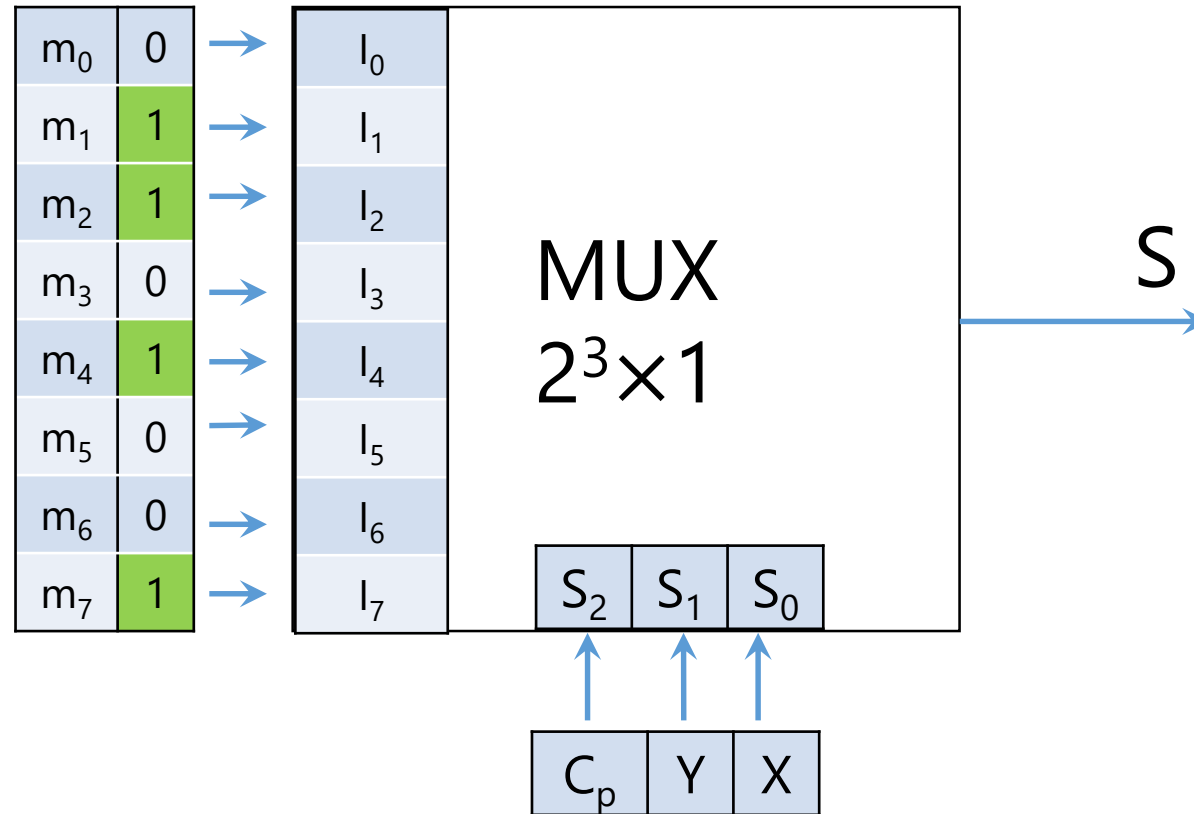
## Full Adder

---

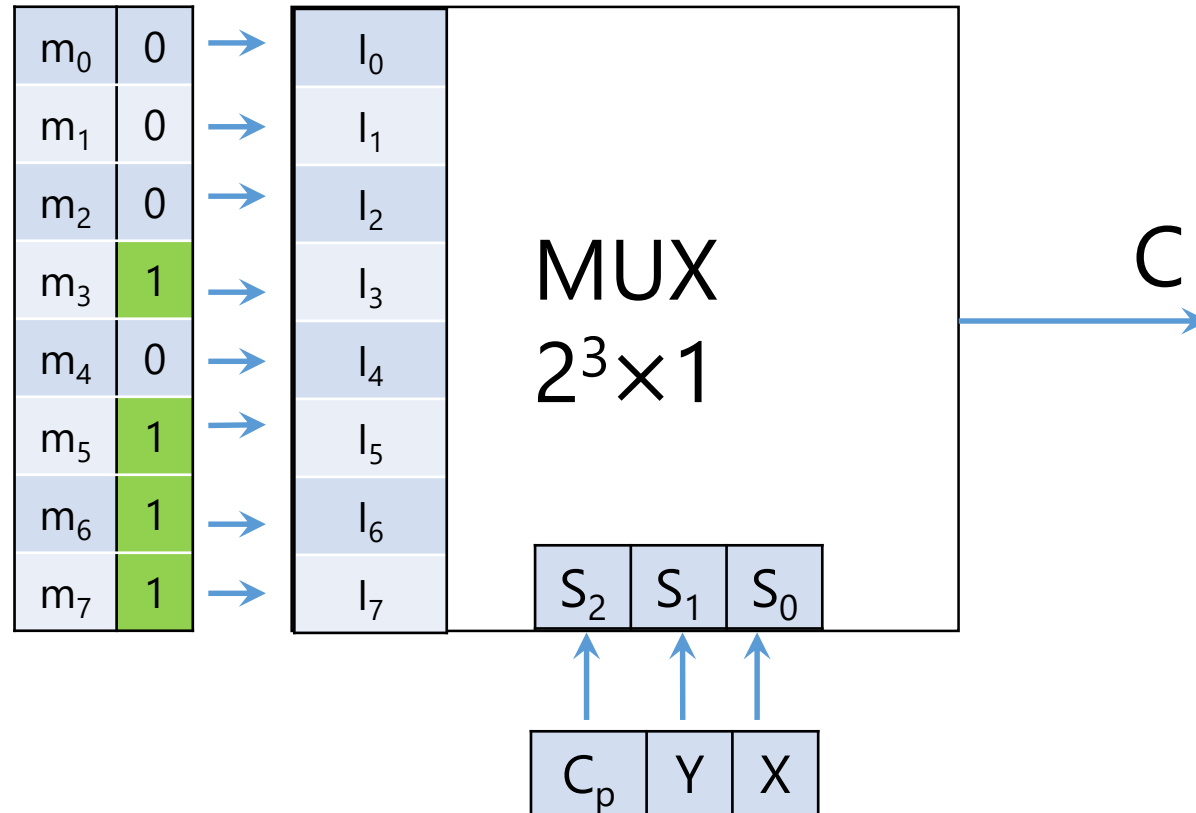
$$S = \sum m(1,2,4,7)$$

$$C = \sum m(3,5,6,7)$$

$C_p$	Y	X	$C = \sum m(3,5,6,7)$	$S = \sum m(1,2,4,7)$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$S = \sum m(1,2,4,7)$$



$$C = \sum m(3,5,6,7)$$

---

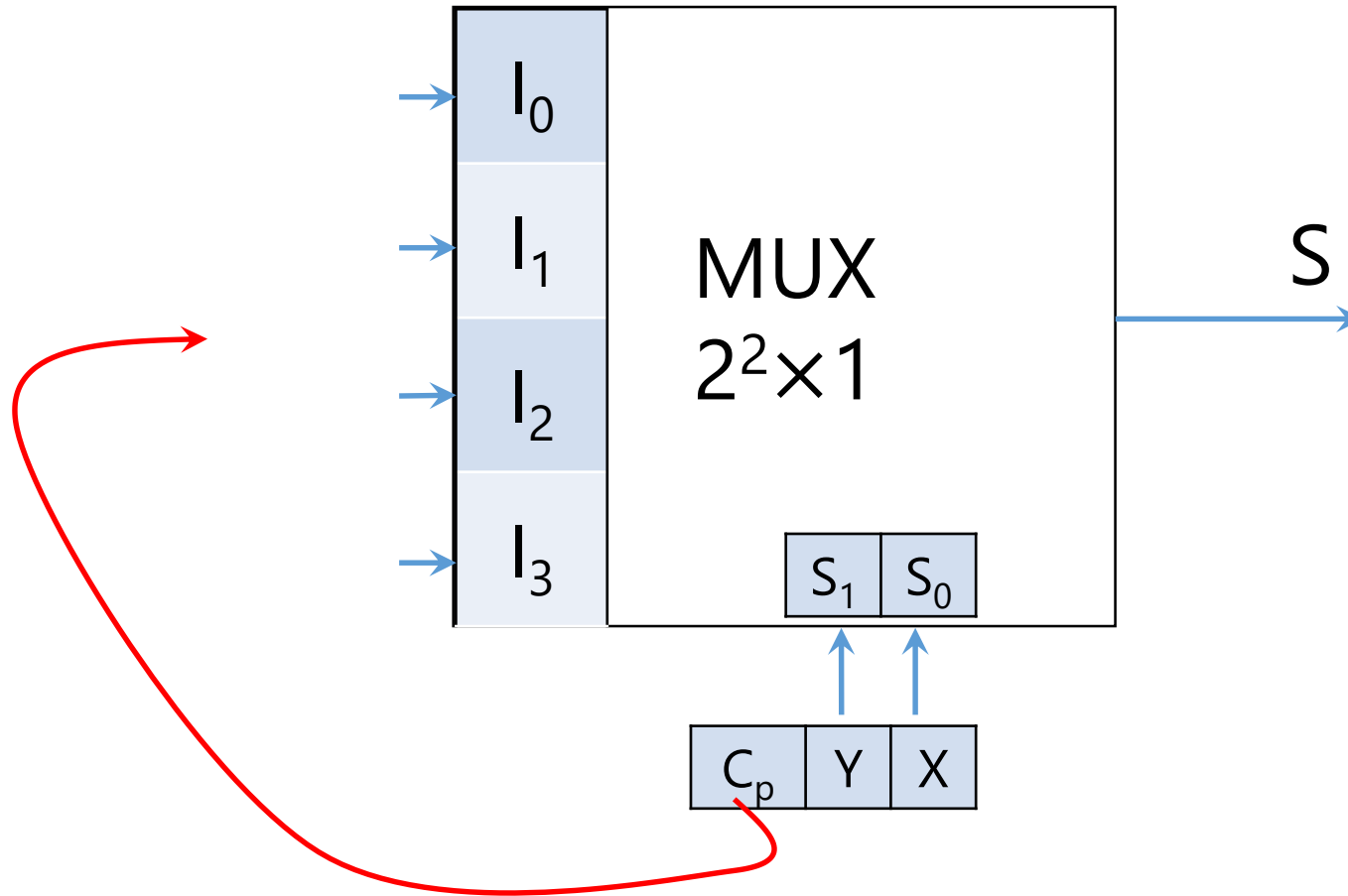
# Multiplexer

## Boolean Function II

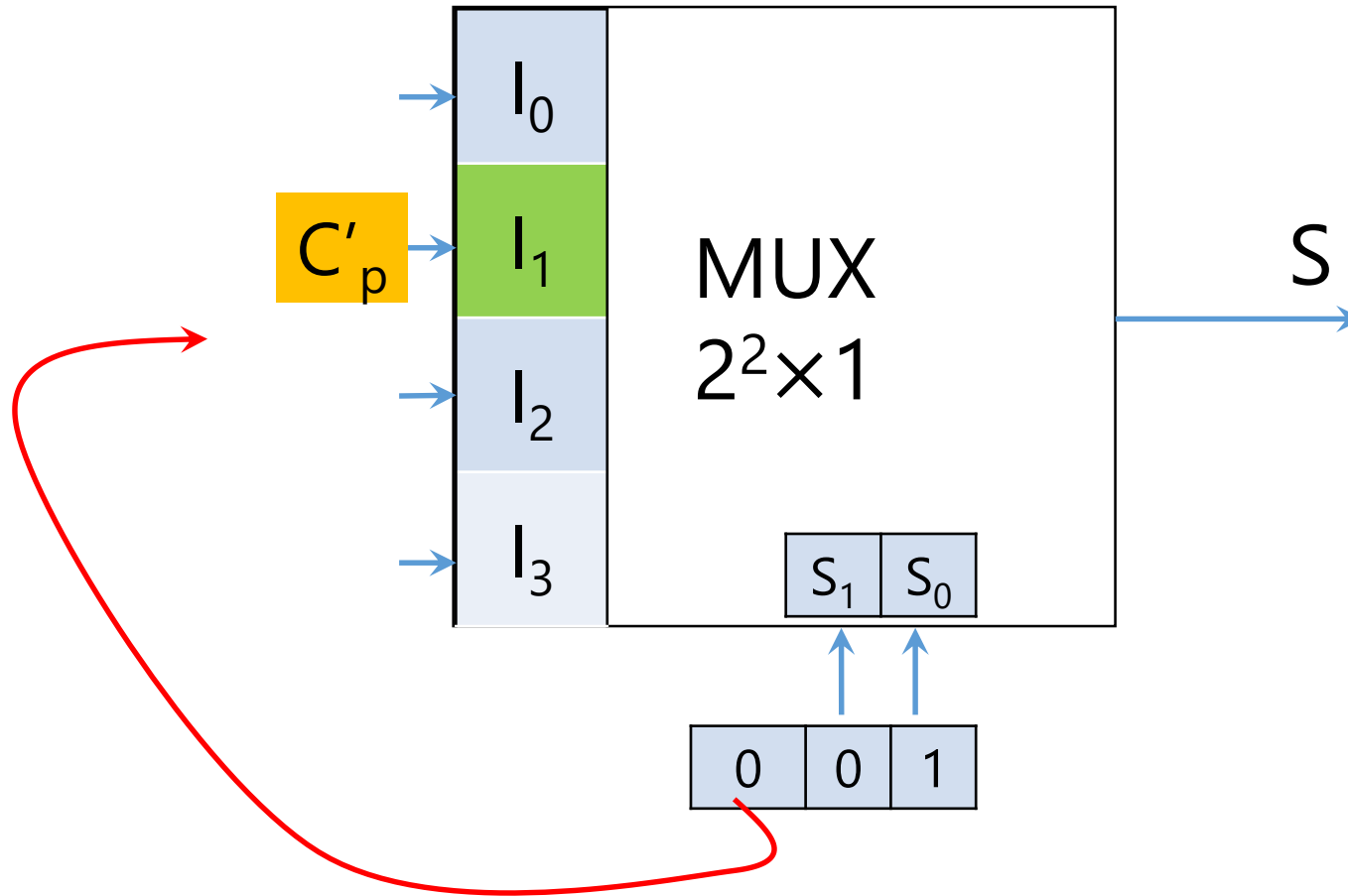
---

Book: Page 161

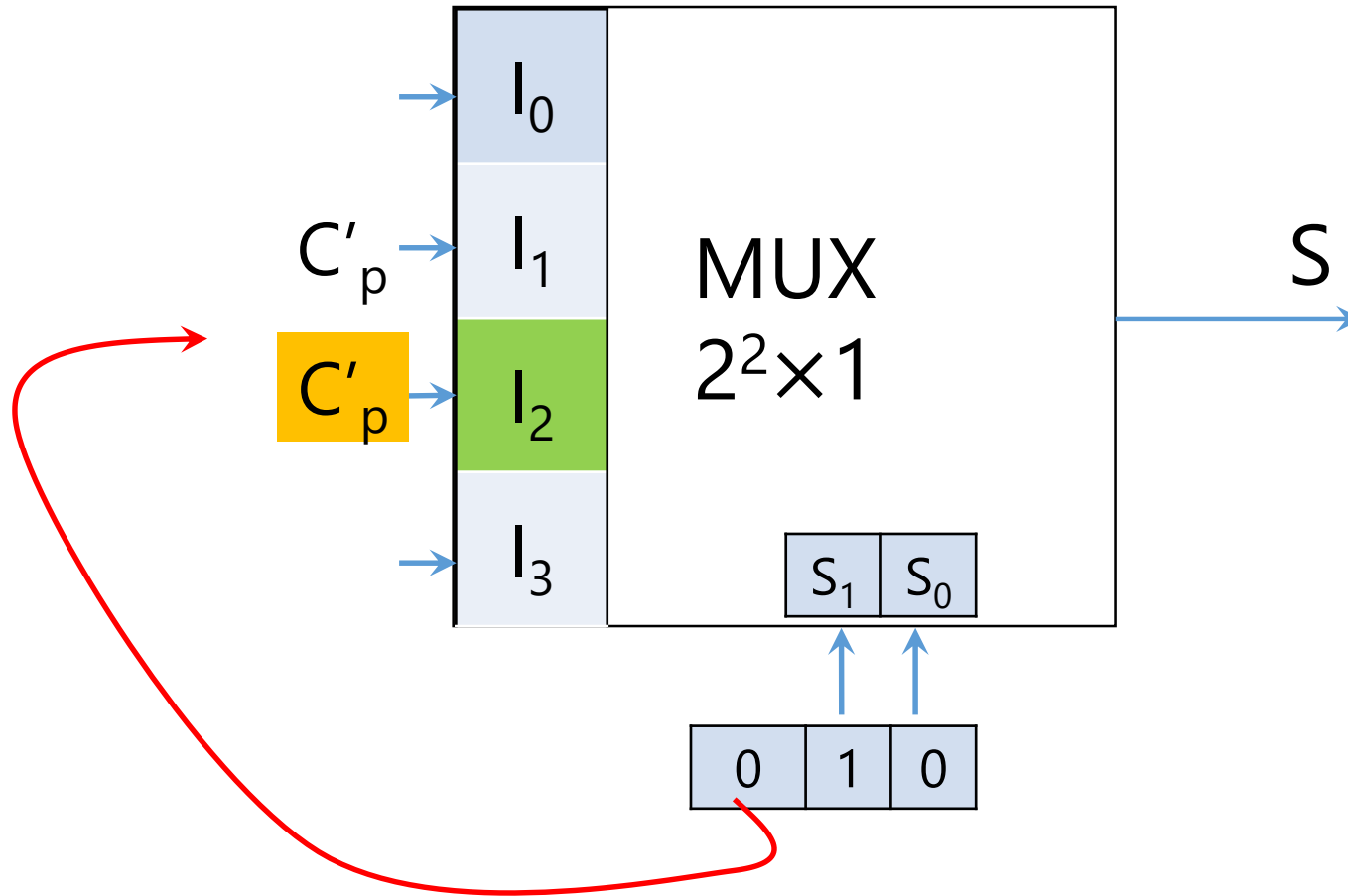




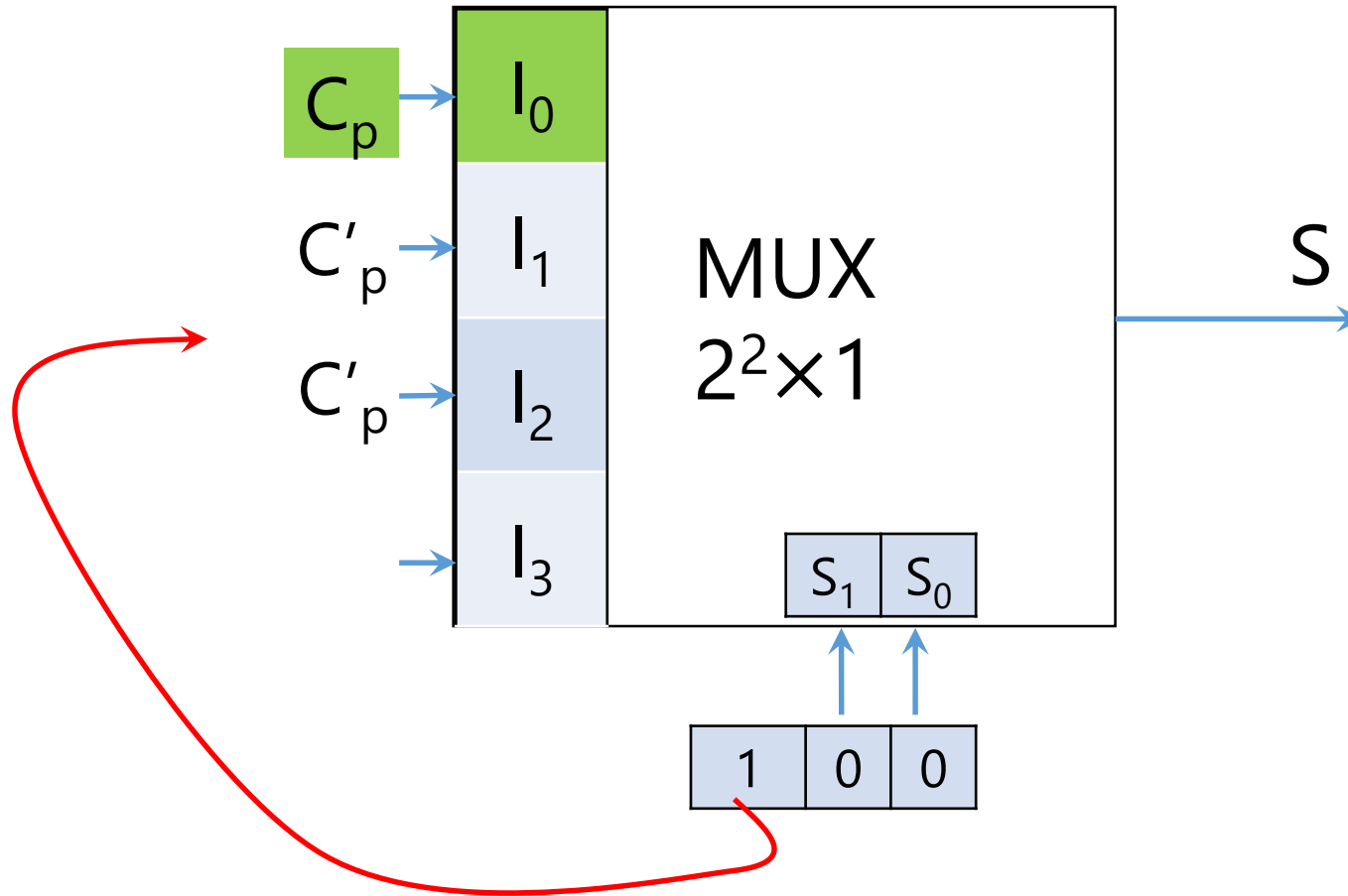
$$S = \sum m(1,2,4,7)$$



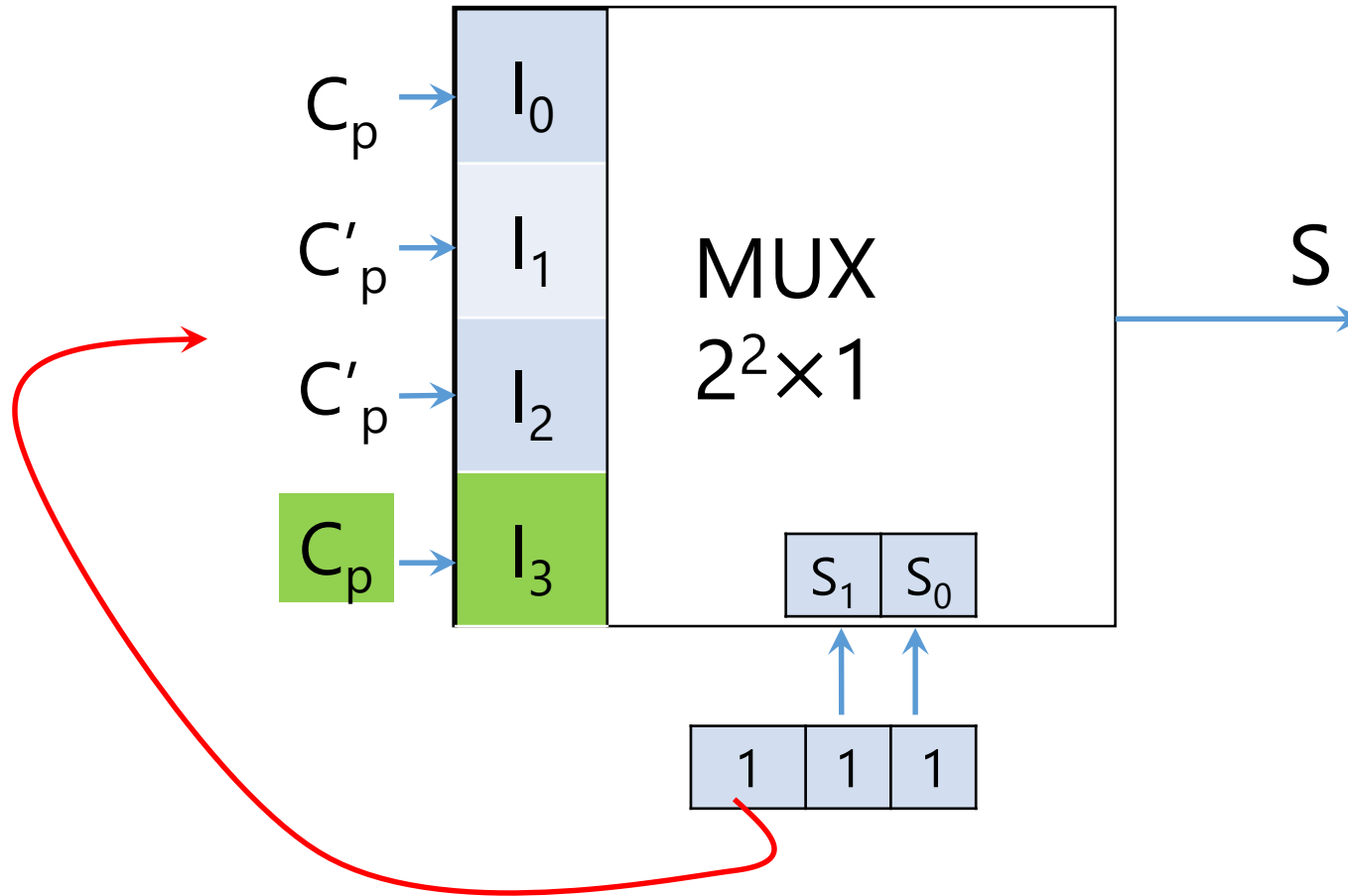
$$S = \sum m(\textcolor{red}{1}, 2, 4, 7)$$



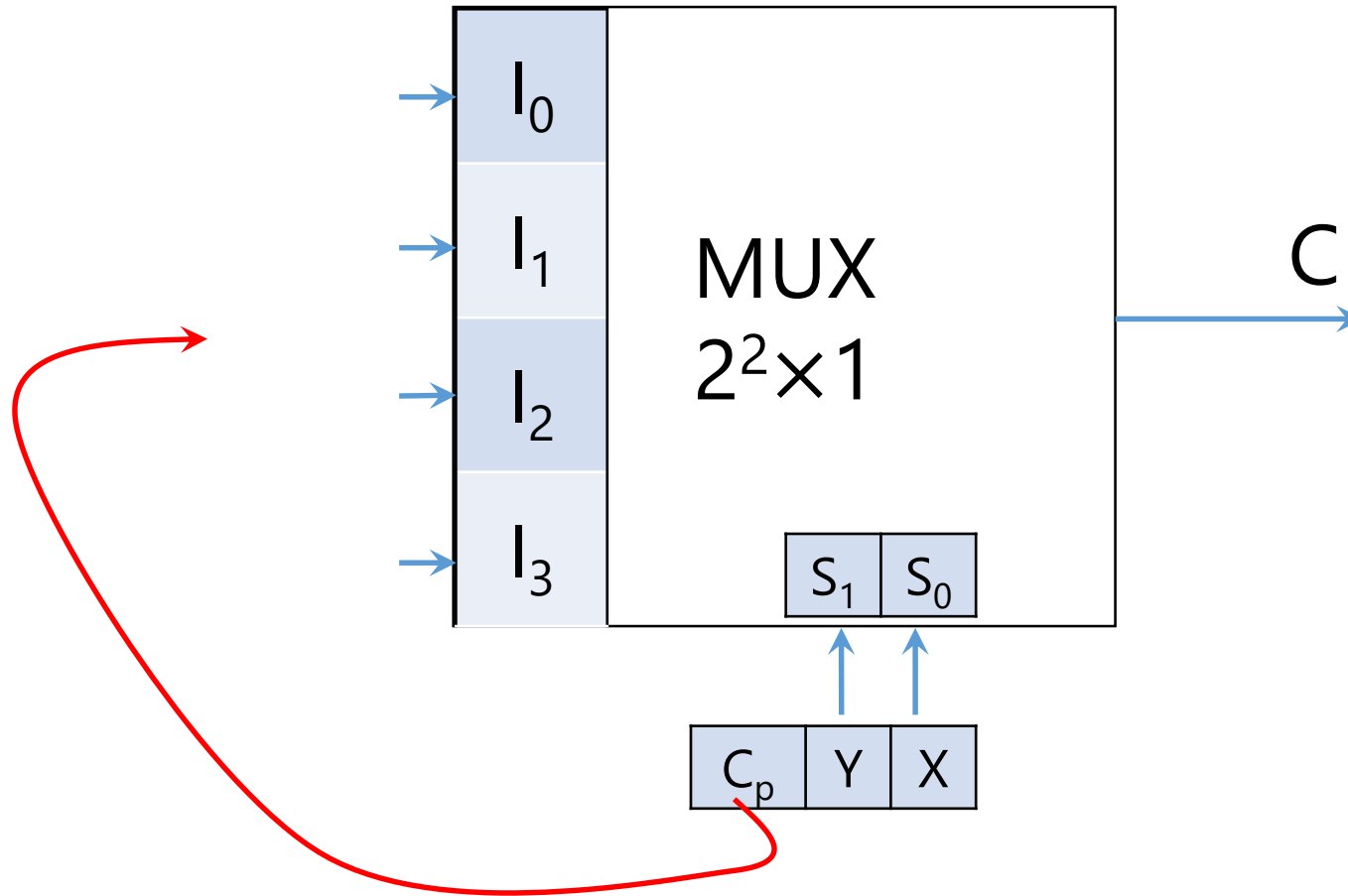
$$S = \sum m(1, \textcolor{red}{2}, 4, 7)$$



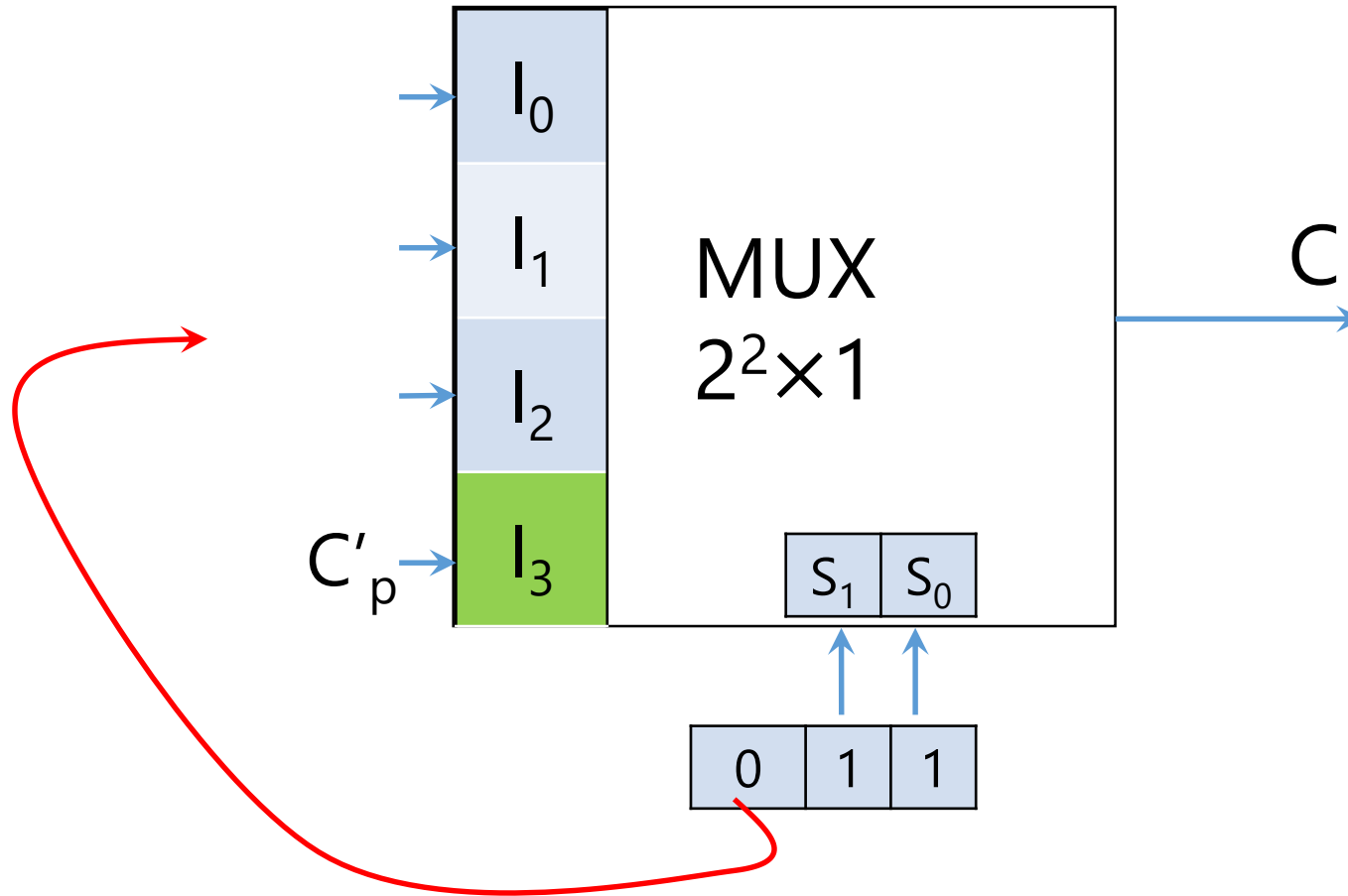
$$S = \sum m(1, 2, \textcolor{red}{4}, 7)$$



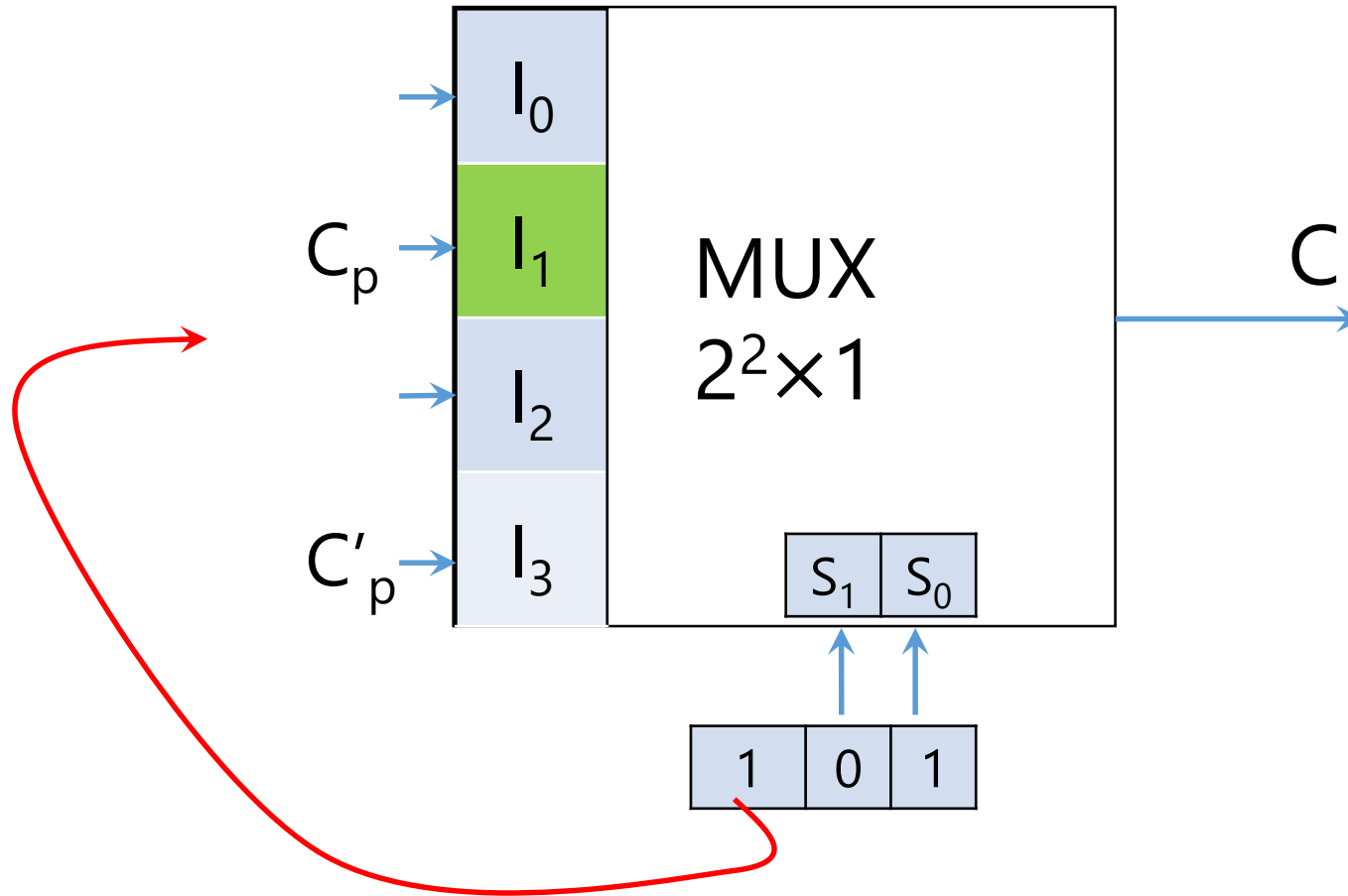
$$S = \sum m(1, 2, 4, \textcolor{red}{7})$$



$$C = \sum m(3, 5, 6, 7)$$

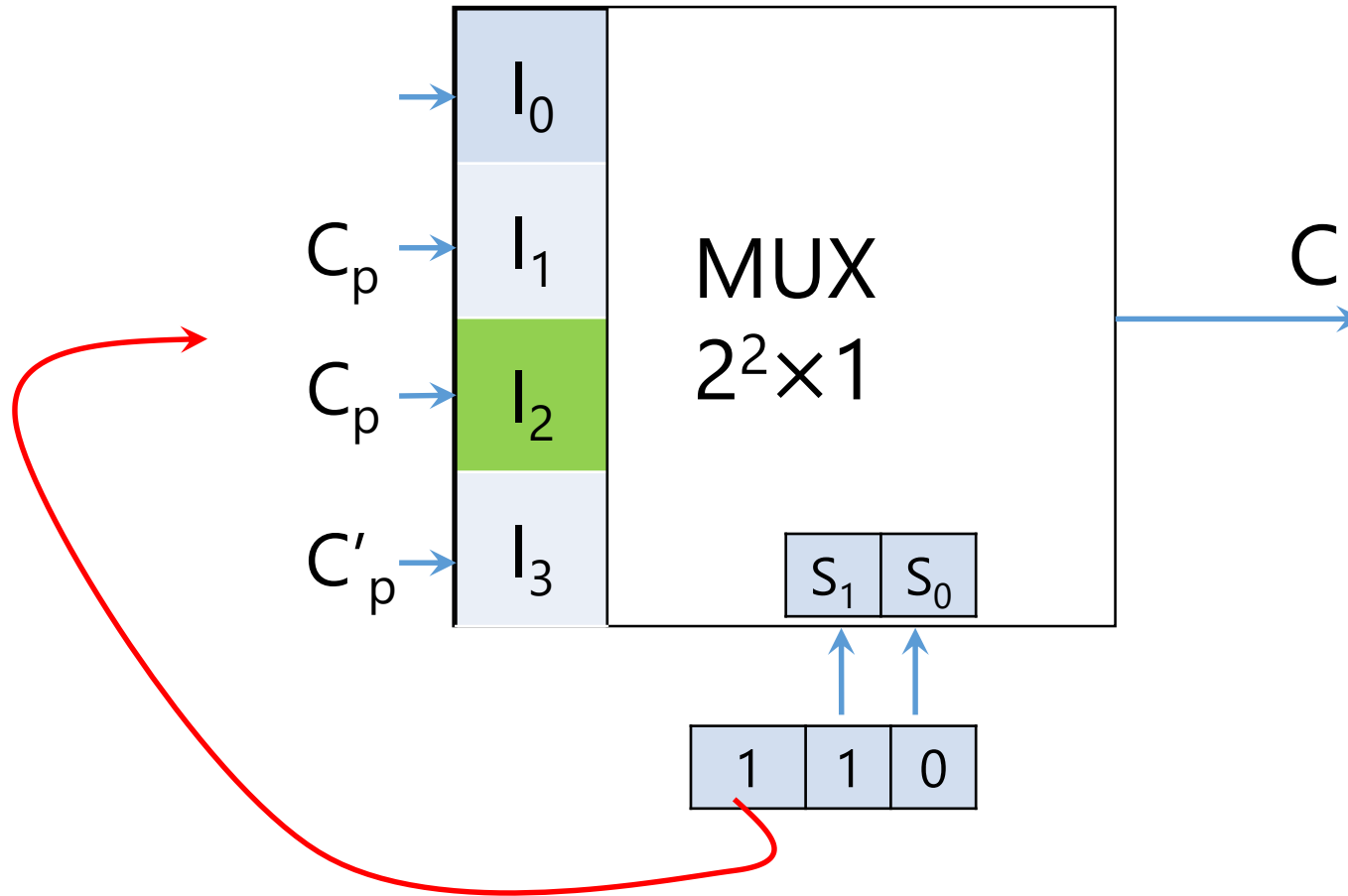


$$C = \sum m(\textcolor{red}{3}, 5, 6, 7)$$

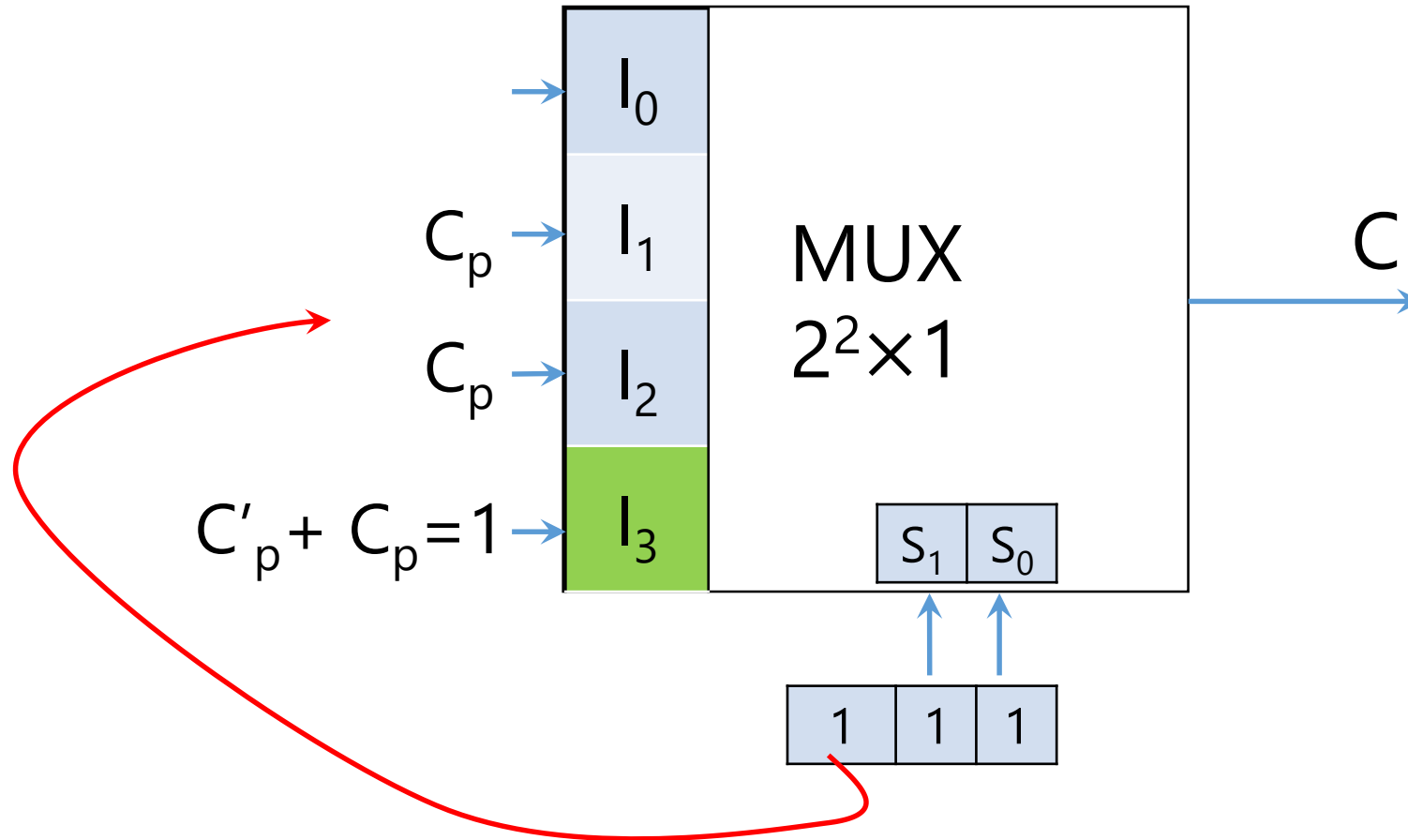


$$C = \sum m(3, \textcolor{red}{5}, 6, 7)$$

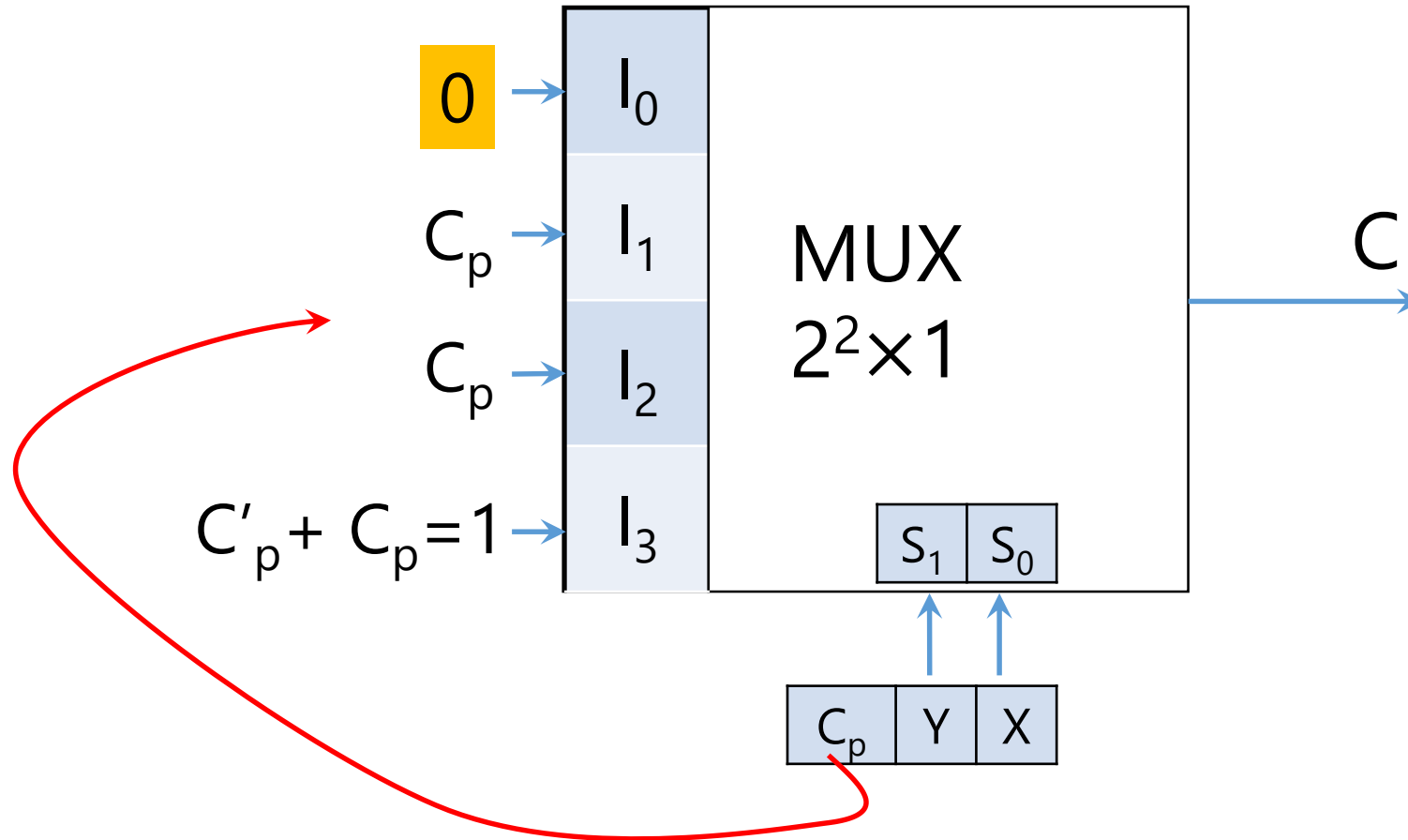




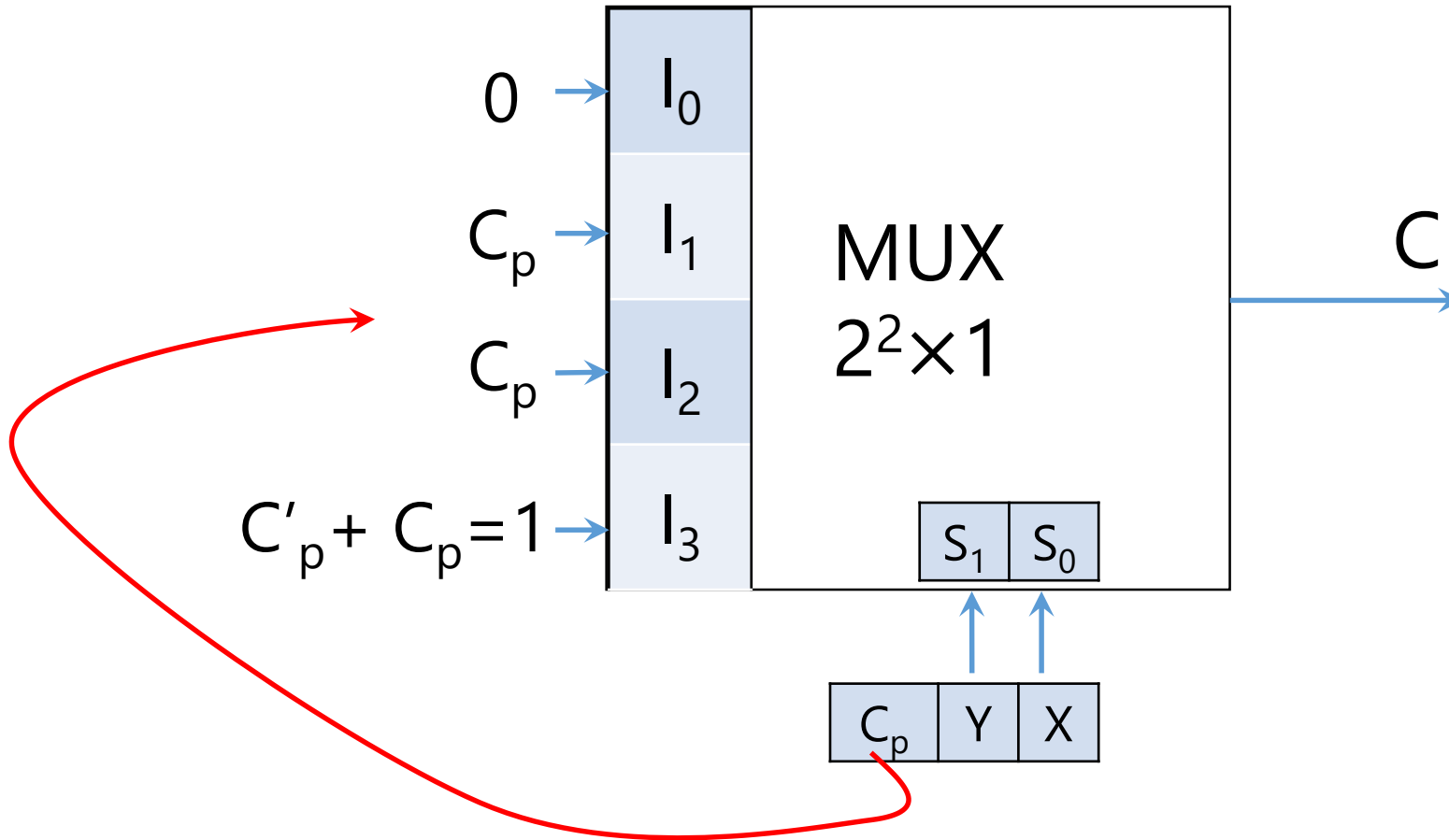
$$C = \sum m(3, 5, \textcolor{red}{6}, 7)$$



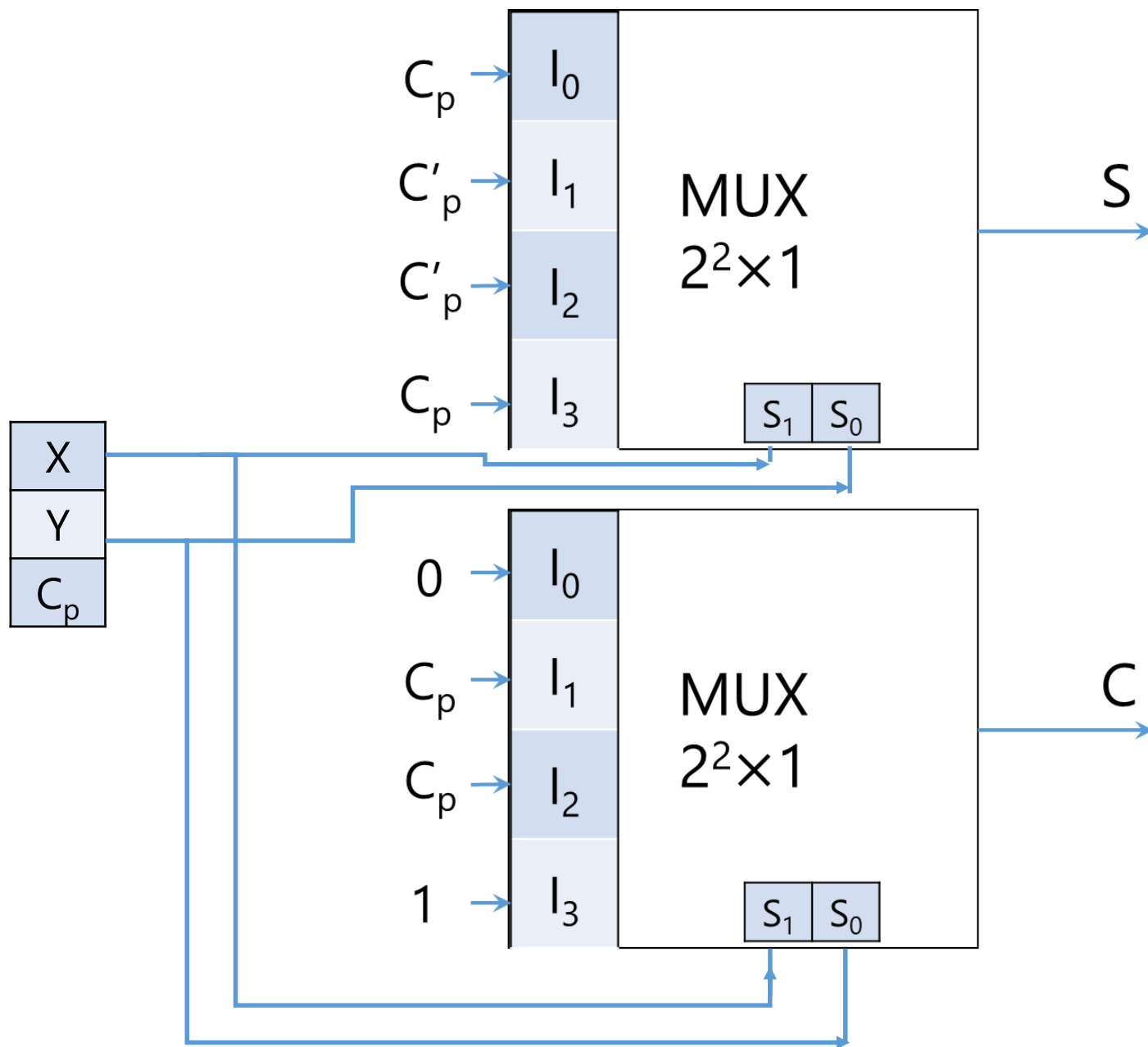
$$C = \sum m(3, 5, 6, 7)$$



$$C = \sum m(3, 5, 6, 7)$$



$$C = \sum m(3, 5, 6, 7)$$



---

# Multiplexer

## Three-State Gates + Decoders

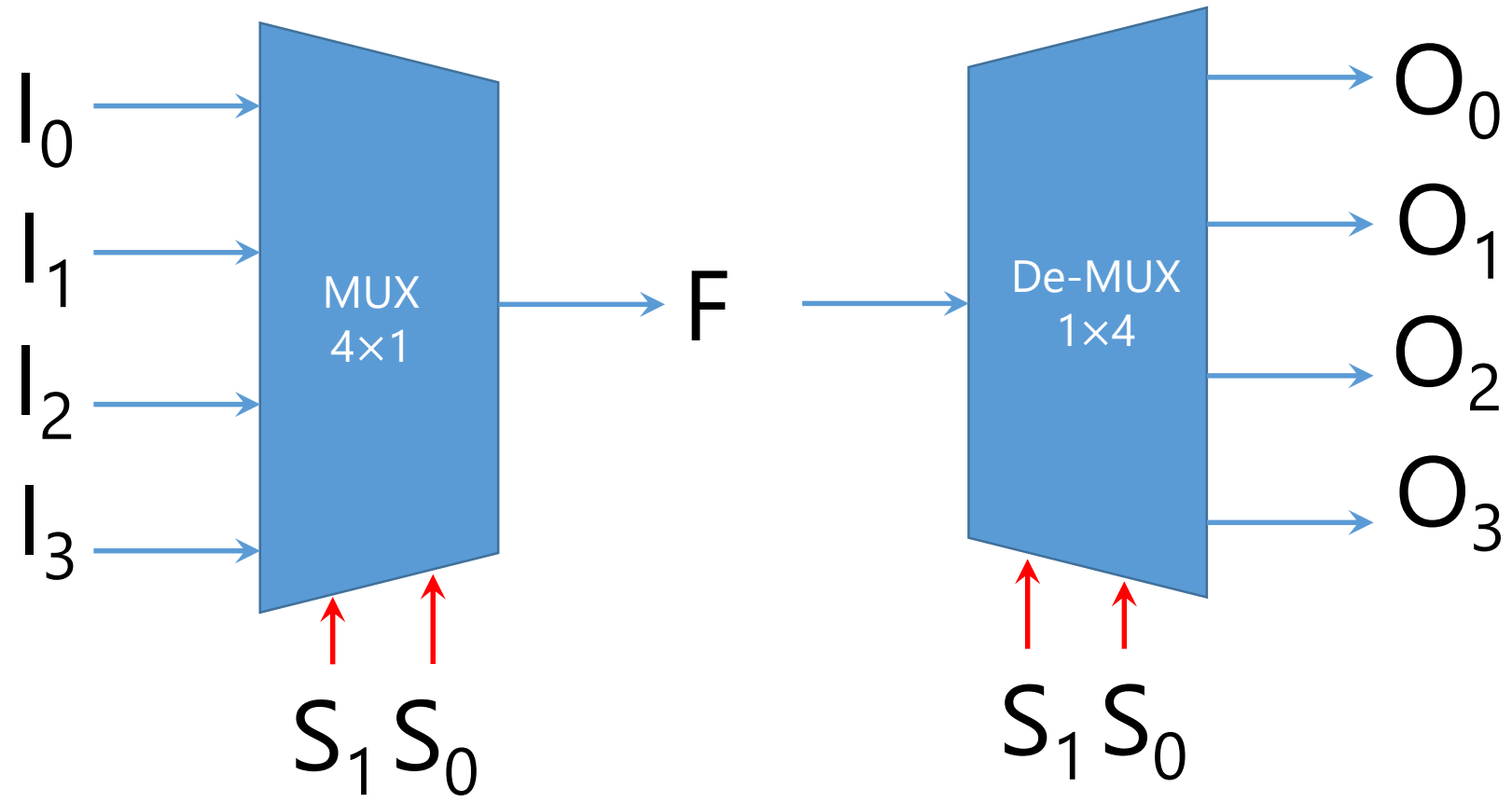
---

Book: Page 162-164

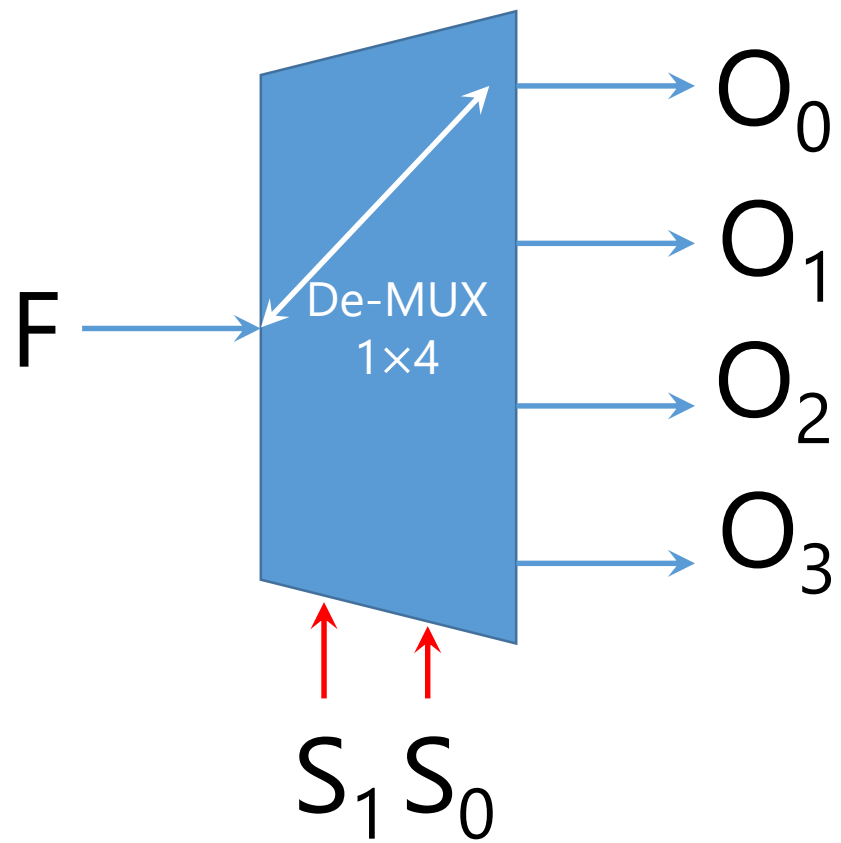
---

# De-multiplexer

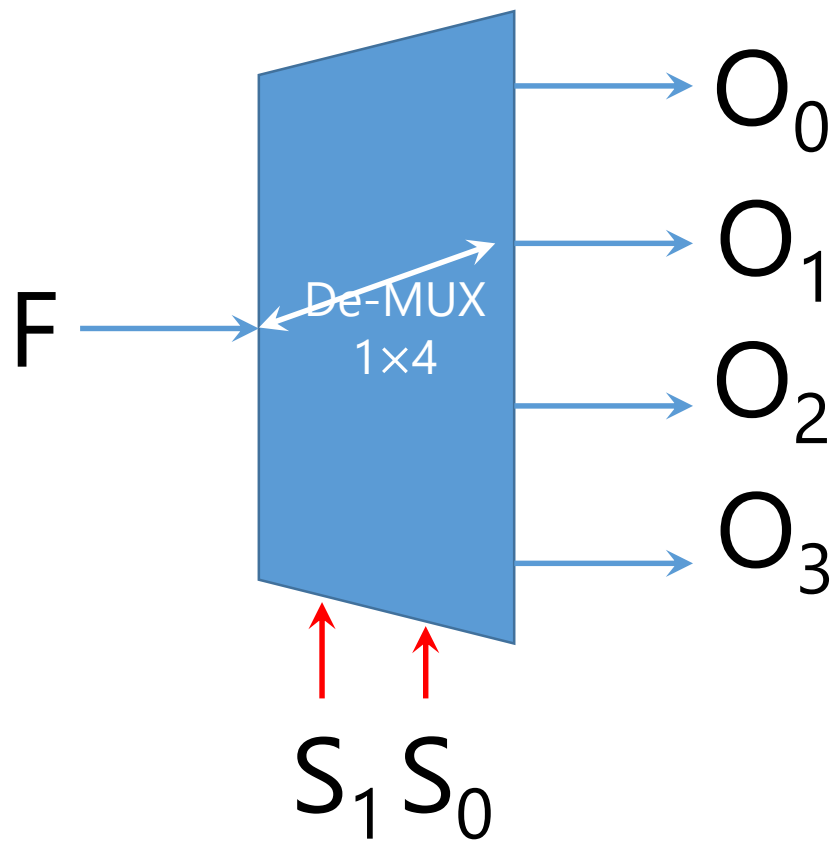
---



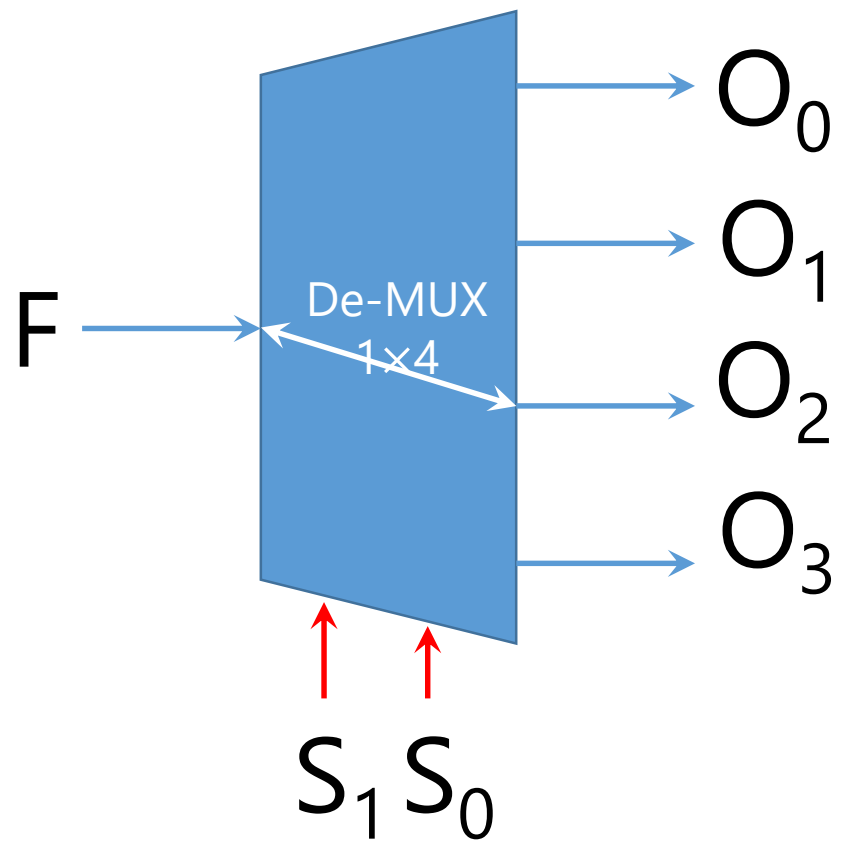




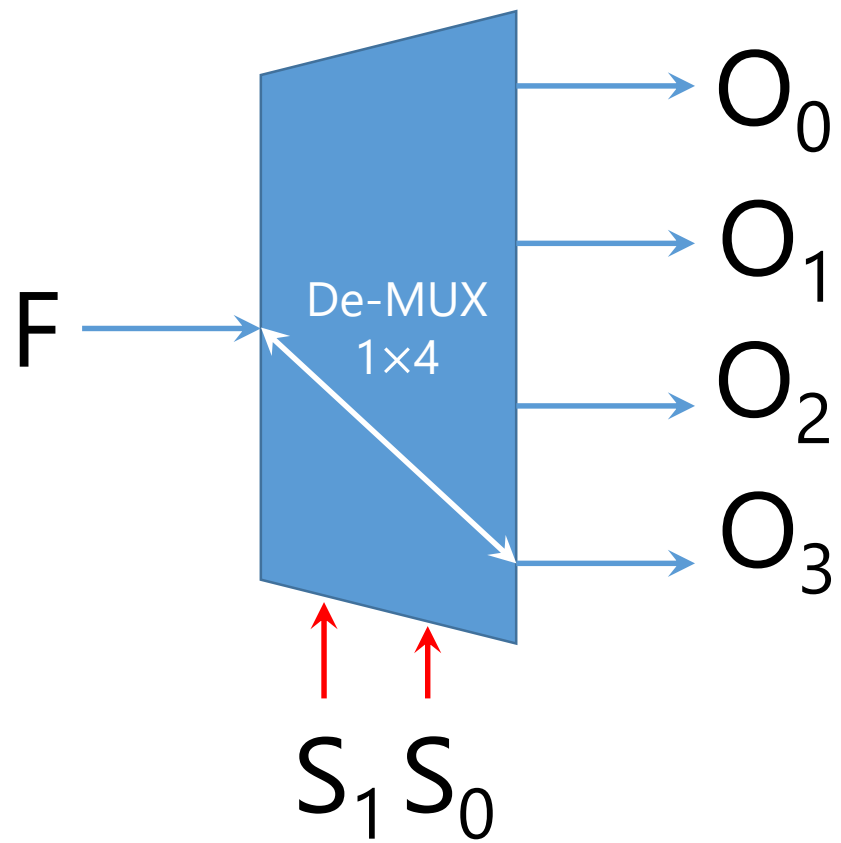
$S_1$	$S_0$	$F$	$O_0$	$O_1$	$O_2$	$O_3$
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1
1	1	1	0	0	0	0



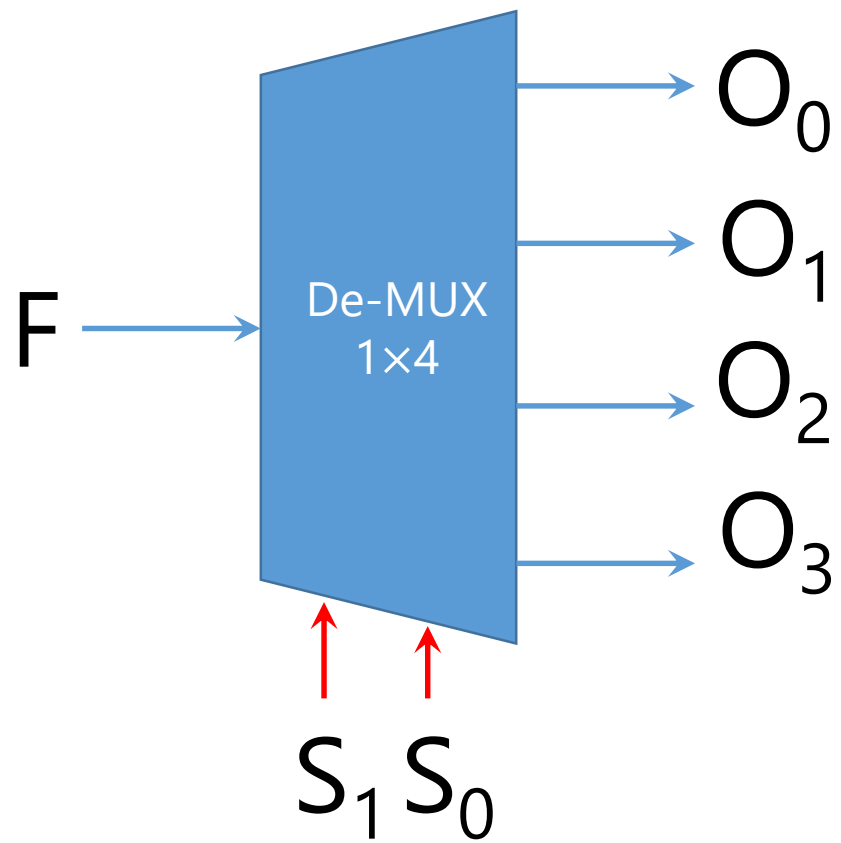
$S_1$	$S_0$	$F$	$O_0$	$O_1$	$O_2$	$O_3$
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1
1	1	1	0	0	0	0



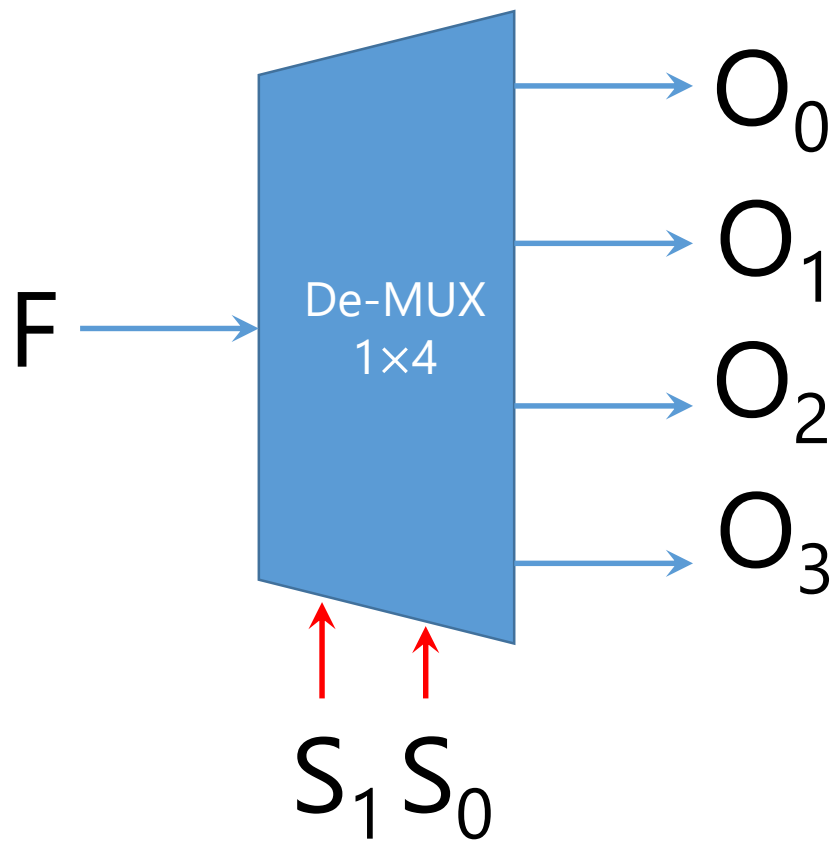
$S_1$	$S_0$	$F$	$O_0$	$O_1$	$O_2$	$O_3$
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1
1	1	1	0	0	0	0



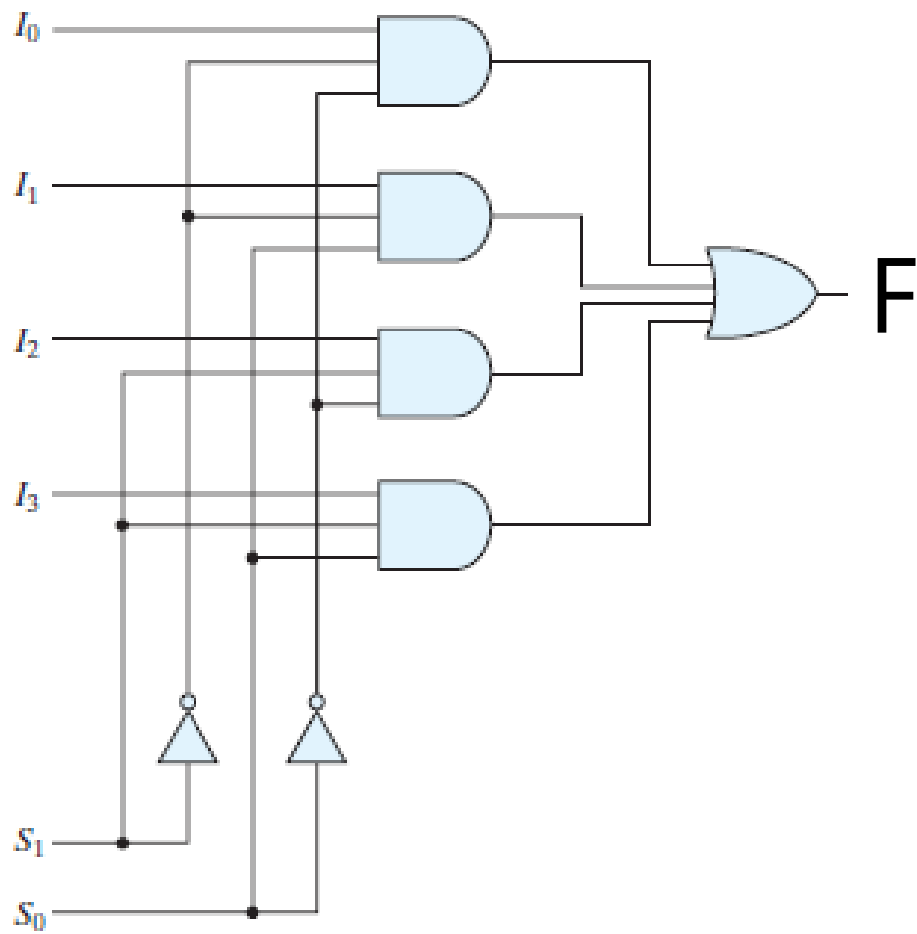
$S_1$	$S_0$	$F$	$O_0$	$O_1$	$O_2$	$O_3$
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1



$S_1$	$S_0$	$O_0$	$O_1$	$O_2$	$O_3$
0	0	F	0	0	0
0	1	0	F	0	0
1	0	0	0	F	0
1	1	0	0	0	F

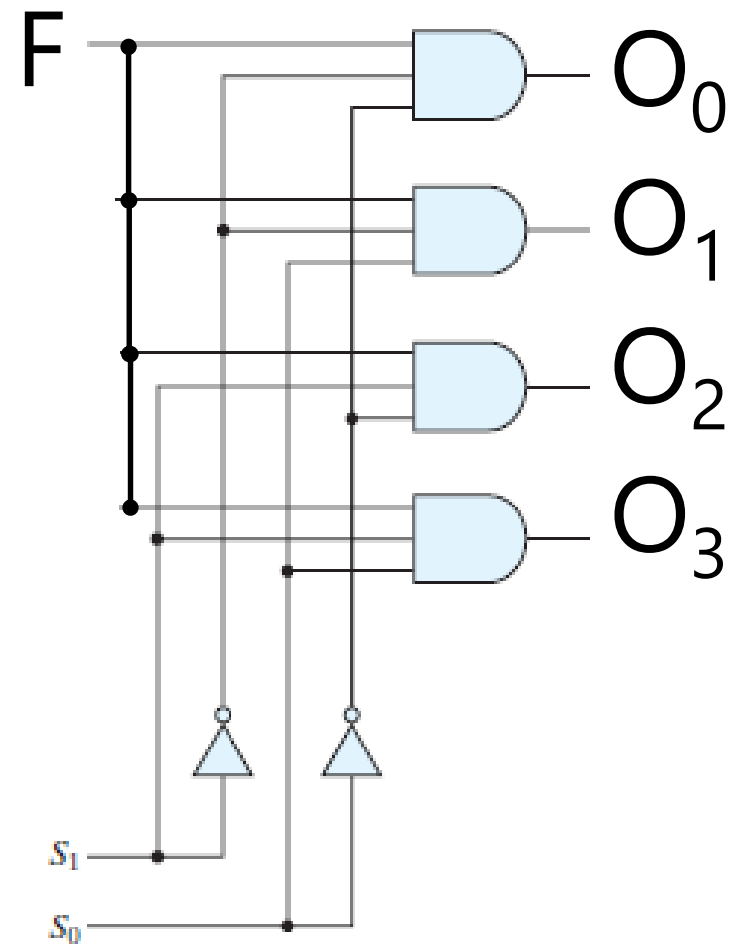


$S_1$	$S_0$	$O_0 = S'_1 S'_0 F$	$O_1 = S'_1 S_0 F$	$O_2 = S_1 S'_0 F$	$O_3 = S_1 S_0 F$
0	0	F	0	0	0
0	1	0	F	0	0
1	0	0	0	F	0
1	1	0	0	0	F



(a) Logic diagram

**FIGURE 4.25**  
Four-to-one-line multiplexer



1-to-4 De-mux

---

De-multiplexer = Decoder w/ Enable input  
How come?!

---



Arithmetic  
&  
Logical Op

Binary Adder, Binary Subtractor, Binary Multiplier

Binary Comparator (Magnitude Comparator)

Data  
Transmission

Decoder, Encoder

Multiplexer (MUX, MPX), De-Multiplexer (Demux)

Coders

**Binary Codes (BCD, Excess-3, Gray)**

---

# Coding

---

---

$A \rightarrow \text{Encode} \rightarrow B$

---

---

$$A \leftarrow \text{Decode} \leftarrow B$$

---

---

$$A \leftrightarrow [\text{En}][\text{De}]\text{code} \leftrightarrow B$$

---

By a convention

- Math, e.g., conversion in radix numbering system
- Non-math, e.g., in base-64, the value of characters
- Engineering
- etc

---

# 1-way Coding

---

$A \rightarrow \text{Encode} \rightarrow B$

$A \xleftarrow{\text{red}} \text{Decode} \xleftarrow{\text{red}} B$

---

# 2-way Coding

---

$A \leftrightarrow \text{Look up Table} \leftrightarrow B$

---

# Base-64

---

$A \leftrightarrow \text{Look up Table} \leftrightarrow B$



Digit	Value		Digit	Value		Digit	Value		Digit	Value
A	0		Q	16		g	32		w	48
B	1		R	17		h	33		x	49
C	2		S	18		i	34		y	50
D	3		T	19		j	35		z	51
E	4		U	20		k	36		0	52
F	5		V	21		l	37		1	53
G	6		W	22		m	38		2	54
H	7	→	X	23	→	n	39	→	3	55
I	8		Y	24		o	40		4	56
J	9		Z	25		p	41		5	57
K	10		a	26		q	42		6	58
L	11		b	27		r	43		7	59
M	12		c	28		s	44		8	60
N	13		d	29		t	45		9	61
O	14		e	30		u	46		+	62
P	15		f	31		v	47		/	63

---

# Binary Codes

Assigning binary numbers to things

---

A  $\leftrightarrow$  Look up Table  $\leftrightarrow$  Binary Number

---

# Binary Coded Decimal

## BCD (8421)

---

Decimal  $\leftrightarrow$  Look up Table  $\leftrightarrow$  Binary Number

**Table 1.4**  
*Binary-Coded Decimal (BCD)*

<b>Decimal Symbol</b>	<b>BCD Digit</b>
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Decimal	BCD (Binary Code)	Binary Number
10	0001 0000	0000 1010
11	0001 0001	0000 1011
12	0001 0010	0000 1100
13	0001 0011	0000 1101
14	0001 0100	0000 1110
15	0001 0101	0000 1111
16	0001 0110	0001 0000
17	0001 0111	0001 0001
18	0001 1000	0001 0010
19	0001 1001	0001 0011
20	0010 0000	0001 0100
21	0010 0001	0001 0101
22	0010 0010	0001 0110
23	0010 0011	0001 0111
...	...	...

Decimal	BCD (Binary Code)	Binary Number
10	0001 0000	0000 1010
11	0001 0001	0000 1011
12	0001 0010	0000 1100
13	0001 0011	0000 1101
14	0001 0100	0000 1110
15	0001 0101	0000 1111
16	0001 0110	0001 0000
17	0001 0111	0001 0001
18	0001 1000	0001 0010
19	0001 1001	0001 0011
20	0010 0000	0001 0100
21	0010 0001	0001 0101
22	0010 0010	0001 0110
23	0010 0011	0001 0111
...	...	...

$$(185)_{10} = (?)_{\text{BCD}} = (?)_2$$

$$(\textcolor{red}{1}85)_{10} = (\textcolor{red}{0001})_{\text{BCD}} = (?)_2$$



$$(1\textcolor{red}{8}5)_{10} = (0001\textcolor{red}{1000})_{\text{BCD}} = (?)_2$$

$$(18\color{red}{5})_{10} = (0001\ 1000\ \color{red}{0101})_{\text{BCD}} = (?)_2$$

$$(185)_{10} = (0001\ 1000\ 0101)_{\text{BCD}} = (?)_2$$

$$(185)_{10} = (0001\ 1000\ 0101)_{\text{BCD}} = (?)_2$$

	Remainder
185 ÷ 2	1
92 ÷ 2	0
46 ÷ 2	0
23 ÷ 2	1
11 ÷ 2	1
5 ÷ 2	1
2 ÷ 2	0
1 ÷ 2	1
0	



$$(185)_{10} = (0001\ 1000\ 0101)_{\text{BCD}} = (10111001)_2$$

---

# Other Binary Codes

---

$A \leftrightarrow \text{Look up Table} \leftrightarrow B$

**Table 1.5***Four Different Binary Codes for the Decimal Digits*

<b>Decimal Digit</b>	<b>BCD 8421</b>	<b>2421</b>	<b>Excess-3</b>	<b>8, 4, -2, -1</b>
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111

---

# Other Binary Codes

## Aiken (2421)

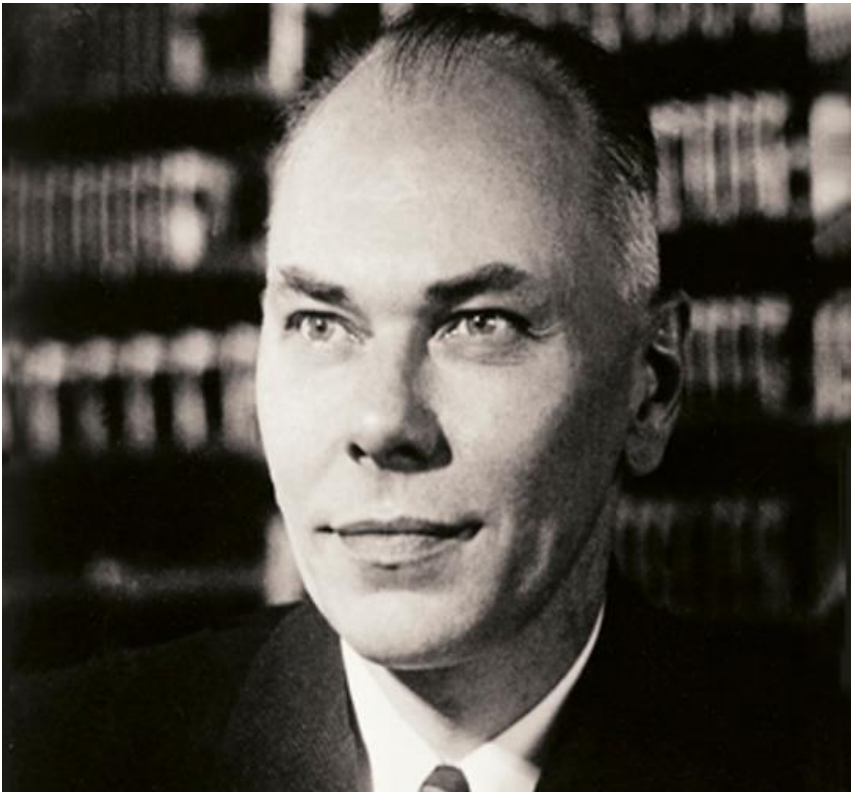
---

*/ˈeɪkən/*

[https://en.wikipedia.org/wiki/Aiken\\_code](https://en.wikipedia.org/wiki/Aiken_code)



		Aiken-Code				
		3	2	1	0	Bit
		2	4	2	1	Wertigkeit
Dezimalzahlen	0					
	1					
	2					
	3					
	4					
Pseudotetraden						
Dezimalzahlen	5					
	6					
	7					
	8					
	9					



# Howard Hathaway Aiken

(March 8, 1900 – March 14, 1973)

Physicist

Pioneer in computing

Original conceptual designer behind IBM's Harvard Mark I

**Table 1.5***Four Different Binary Codes for the Decimal Digits*

<b>Decimal Digit</b>	<b>BCD 8421</b>	<b>Aiken 2421</b>	<b>Excess-3</b>	<b>8, 4, -2, -1</b>
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111

The diagram illustrates the relationship between the Aiken 2421 and Excess-3 codes. Blue arrows show the mapping from Aiken to Excess-3: for digits 0-3, the Aiken code is shifted 1 unit to the right to get the Excess-3 code; for digits 4-9, the Aiken code is shifted 3 units to the right to get the Excess-3 code. A red 'NOT' label is placed near the transition between digits 4 and 5, indicating that the shift operation is not a simple 1-unit shift for all digits.

$$\begin{aligned}(185)_{10} &= (0001\ 1000\ 0101)_{\text{BCD (8421)}} \\ &= (10111001)_2 \\ &= (0001\ \text{NOT}(1)\ \text{NOT}(4))_{\text{Aiken (2421)}}\end{aligned}$$

$$\begin{aligned}(185)_{10} &= (0001\ 1000\ 0101)_{\text{BCD (8421)}} \\ &= (10111001)_2 \\ &= (0001\ \text{NOT}(0001)\ \text{NOT}(0100))_{\text{Aiken (2421)}}\end{aligned}$$

$$\begin{aligned}(185)_{10} &= (0001\ 1000\ 0101)_{\text{BCD (8421)}} \\ &= (10111001)_2 \\ &= (0001\ 1110\ 1011)_{\text{Aiken (2421)}}\end{aligned}$$

---

# Other Binary Codes

## Excess-3 (XS-3)

---

<https://en.wikipedia.org/wiki/Excess-3>

## George Robert Stibitz

(April 30, 1904 – January 31, 1995)

Bell Labs researcher

One of the fathers of the modern first digital computer

**Table 1.5**

*Four Different Binary Codes for the Decimal Digits*

Decimal Digit	BCD 8421	2421	<b>+3</b> Excess-3	8, 4, -2, -1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111



$$\begin{aligned}(185)_{10} &= (0001\ 1000\ 0101)_{\text{BCD (8421)}} \\ &= (10111001)_2 \\ &= (0001\ 1110\ 1011)_{\text{Aiken (2421)}} \\ &= ((1+3)\ (8+3)\ (5+3))_{\text{Excess-3}}\end{aligned}$$



$$\begin{aligned}(185)_{10} &= (0001\ 1000\ 0101)_{\text{BCD (8421)}} \\ &= (10111001)_2 \\ &= (0001\ 1110\ 1011)_{\text{Aiken (2421)}} \\ &= ((4)\ (1\ 1)\ (8))_{\text{Excess-3}}\end{aligned}$$

$$\begin{aligned}(185)_{10} &= (0001\ 1000\ 0101)_{\text{BCD (8421)}} \\ &= (10111001)_2 \\ &= (0001\ 1110\ 1011)_{\text{Aiken (2421)}} \\ &= (\textcolor{red}{0100}\ \textcolor{red}{1011}\ \textcolor{red}{1000})_{\text{Excess-3}}\end{aligned}$$

---

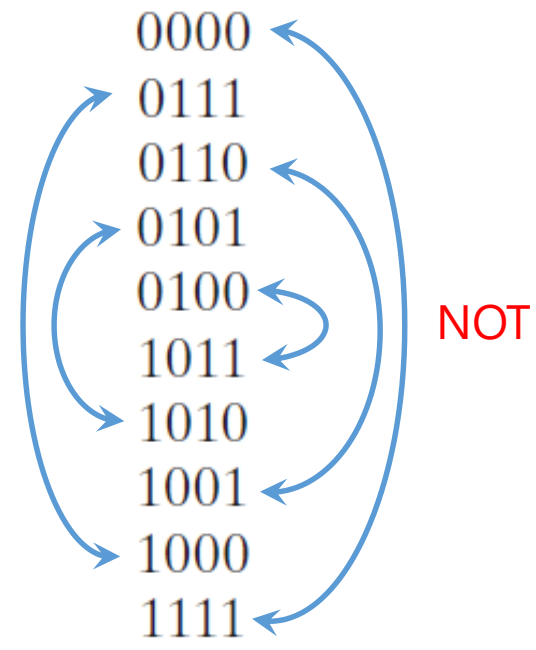
# Other Binary Codes

84(-2)(-1)

---

**Table 1.5***Four Different Binary Codes for the Decimal Digits*

Decimal Digit	BCD 8421	Aiken 2421	Excess-3	8, 4, -2, -1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111



$$\begin{aligned}
(185)_{10} &= (0001\ 1000\ 0101)_{\text{BCD (8421)}} \\
&= (10111001)_2 \\
&= (0001\ 1110\ 1011)_{\text{Aiken (2421)}} \\
&= (0100\ 1011\ 1000)_{\text{Excess-3}} \\
&= (\textcolor{red}{0111}\ \textcolor{red}{1000}\ \textcolor{red}{1011})_{84-2-1}
\end{aligned}$$

---

What's nice about *some* binary codes?

---

---

## Self-complementing

---

The 9's complement of the decimal number  
=  
The 1's complement (NOT) of its binary code

$$\begin{aligned}
 (185)_{10} &= (0001\ 1110\ 1011)_{\text{Aiken (2421)}} \\
 &= (0100\ 1011\ 1000)_{\text{Excess-3}} \\
 &= (0111\ 1000\ 1011)_{84-2-1}
 \end{aligned}$$

$$\begin{aligned}
 9'\text{s-comp}(185)_{10} &= (814)_{10} \\
 &= \text{NOT}(0001\ 1110\ 1011)_{\text{Aiken (2421)}} \\
 &= \text{NOT}(0100\ 1011\ 1000)_{\text{Excess-3}} \\
 &= \text{NOT}(0111\ 1000\ 1011)_{84-2-1}
 \end{aligned}$$



$$\begin{aligned}
 (185)_{10} &= (0001\ 1110\ 1011)_{\text{Aiken (2421)}} \\
 &= (0100\ 1011\ 1000)_{\text{Excess-3}} \\
 &= (0111\ 1000\ 1011)_{84-2-1}
 \end{aligned}$$

$$\begin{aligned}
 9'\text{'s-comp}(185)_{10} &= (814)_{10} \\
 &= (1110\ 0001\ 0100)_{\text{Aiken (2421)}} \\
 &= (1011\ 0100\ 0111)_{\text{Excess-3}} \\
 &= (1000\ 0111\ 0100)_{84-2-1}
 \end{aligned}$$

---

# Other Binary Codes

## Gray

---

**Table 1.6**  
*Gray Code*

<b>Gray Code</b>	<b>Decimal Equivalent</b>
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

---

# Gray Code

## Analog → Digital

---

**Table 1.6**  
*Gray Code*

Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

---

# Gray Code

## Analog → Digital

---

***Straight binary*** number sequence for 7 to 8: 0111 → 1000; causes all four bits to change values.

***Gray*** code for 7 → 8: 0100 to 1100; only the first bit changes from 0 to 1; the other three bits remain the same.

---

# Gray Code Algorithm

---

Step 0: Convert the decimal number to binary number.

Step 1: The MSB (Most Significant Bit) of a gray code and binary code is **the same**.

Step 2: The next digit of gray code is the XOR of the previous and current digit in the binary code.

Step 0: Convert the decimal number to binary number.

(20) <sub>10</sub>	Binary Number	1	0	1	0	0
	Gray Code					



Step 1: The MSB (Most Significant Bit) of a gray code and binary code is **the same**.

(20) <sub>10</sub>	Binary Number	1	0	1	0	0
	Gray Code	1				

Step 2: The next digit of gray code is the XOR of the previous and current digit in the binary code.

(20) <sub>10</sub>	Binary Number	1	0	1	0	0
	Gray Code	1	$1 \oplus 0 = 1$			

Step 2: The next digit of gray code is the XOR of the previous and current digit in the binary code.

(20) <sub>10</sub>	Binary Number	1	0	1	0	0
	Gray Code	1	1	$0 \oplus 1 = 1$		

Step 2: The next digit of gray code is the XOR of the previous and current digit in the binary code.

(20) <sub>10</sub>	Binary Number	1	0	1	0	0
	Gray Code	1	1	1	$1 \oplus 0 = 1$	

Step 2: The next digit of gray code is the XOR of the previous and current digit in the binary code.

(20) <sub>10</sub>	Binary Number	1	0	1	0	0
	Gray Code	1	1	1	1	$0 \oplus 0 = 0$

Step 2: The next digit of gray code is the XOR of the previous and current digit in the binary code.

(20) <sub>10</sub>	Binary Number	1	0	1	0	0
	Gray Code	1	1	1	1	0

(21) <sub>10</sub>	Binary Number	1	0	1	0	1
	Gray Code	1	1	1	1	1

---

# ASCII Code

**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

---

# USASCII code chart

<div> <div> b<sub>7</sub> b<sub>6</sub> b<sub>5</sub> </div> <div> b<sub>4</sub> b<sub>3</sub> b<sub>2</sub> b<sub>1</sub> </div> <div> Column Row </div> </div>					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
					0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	\$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL



$$\text{"0"} = (011\ 0000)_2 = (48)_{10}$$

**Table 1.7**

*American Standard Code for Information Interchange (ASCII)*

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

**Control Characters**

NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

---

# Combinational Logic

## Binary Codes

---

---

# Combinational Logic

## Code Conversion

---

Decimal Equivalent	Aiken				Gray Code
	BCD 8421	2421	Excess-3	8, 4, -2, -1	
0	0000	0000	0011	0000	0000
1	0001	0001	0100	0111	0001
2	0010	0010	0101	0110	0011
3	0011	0011	0110	0101	0010
4	0100	0100	0111	0100	0110
5	0101	1011	1000	1011	0111
6	0110	1100	1001	1010	0101
7	0111	1101	1010	1001	0100
8	1000	1110	1011	1000	1100
9	1001	1111	1100	1111	1101
10					1111
11					1110
12					1010
13					1011
14					1001
15					1000

Decimal Equivalent	BCD 8421	Aiken 2421	Excess-3	8, 4, -2, -1	Gray Code
0	0000	0000	0011	0000	0000
1	0001	0001	0100	0111	0001
2	0010	0010	0101	0110	0011
3	0011	0011	0110	0101	0010
4	0100	0100	0111	0100	0110
5	0101	1011	1000	1011	0111
6	0110	1100	1001	1010	0101
7	0111	1101	1010	1001	0100
8	1000	1110	1011	1000	1100
9	1001	1111	1100	1111	1101
10	0001 0000	0001 0000	You fill it at home	You fill it at home	1111
11	0001 0001	0001 0001			1110
12	0001 0010	0001 0010			1010
13	0001 0011	0001 0011			1011
14	0001 0100	0001 0100			1001
15	0001 0101	0001 1011			1000

Decimal Equivalent	<b>Aiken</b>				Gray Code
	BCD 8421	2421	Excess-3	8, 4, -2, -1	
0	0000	0000	0011	0000	0000
1	0001	0001	0100	0111	0001
2	0010	0010	0101	0110	0011
3	0011	0011	0110	0101	0010
4	0100	0100	0111	0100	0110
5	0101	1011	1000	1011	0111
6	0110	1100	1001	1010	0101
7	0111	1101	1010	1001	0100
8	1000	1110	1011	1000	1100
9	1001	1111	1100	1111	1101
10	0001 0000	0001 0000	You fill it at home	You fill it at home	1111
11	0001 0001	0001 0001			1110
12	0001 0010	0001 0010			1010
13	0001 0011	0001 0011			1011
14	0001 0100	0001 0100			1001
15	0001 0101	0001 1011			1000

---

# Combinational Logic

## Code Conversion

---

BCD (8421)  $\rightarrow$  Excess-3

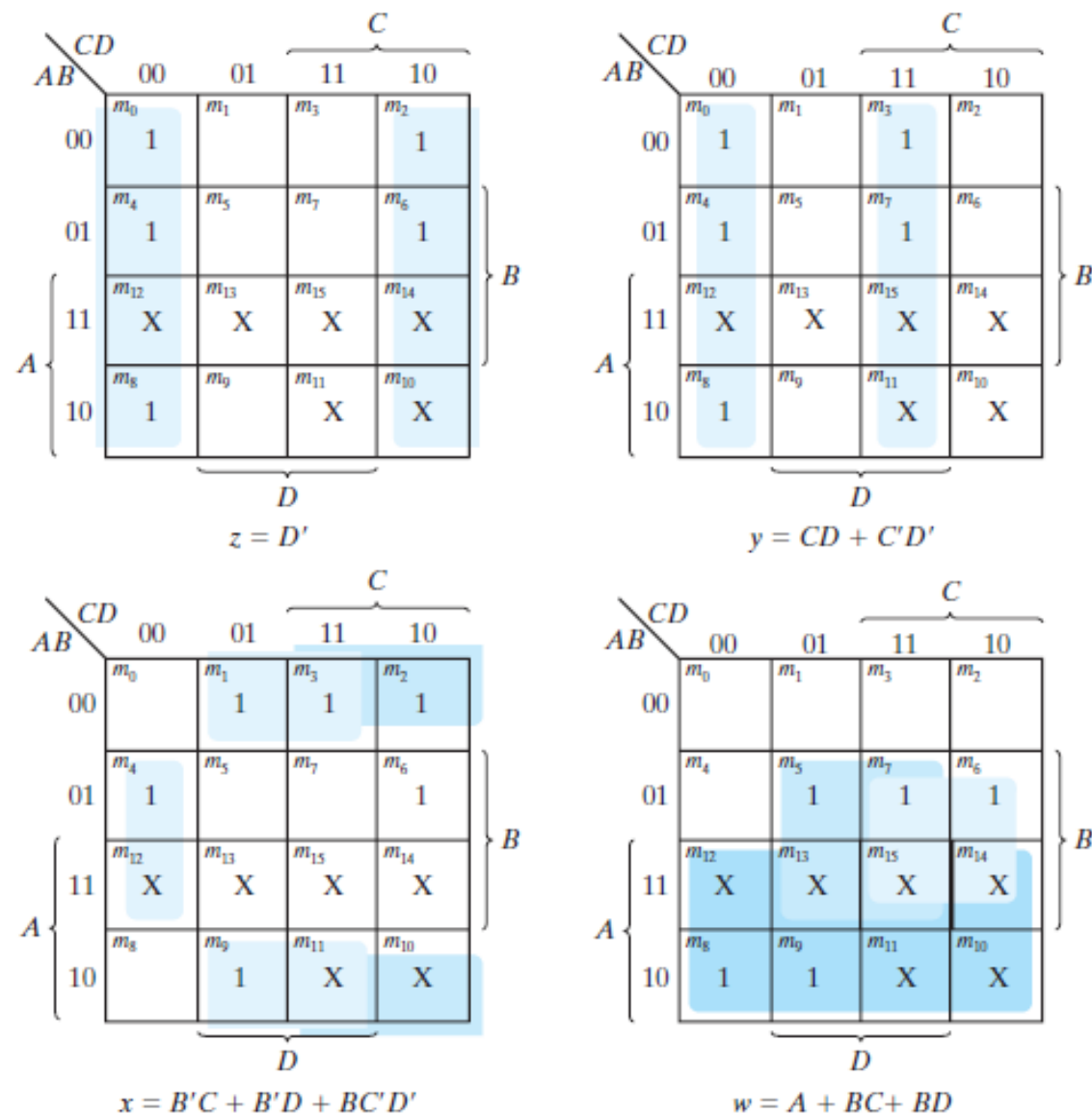
**Table 4.2***Truth Table for Code Conversion Example*

Input BCD				Output Excess-3 Code			
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>w</i>	<i>x</i>	<i>y</i>	<i>z</i>
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

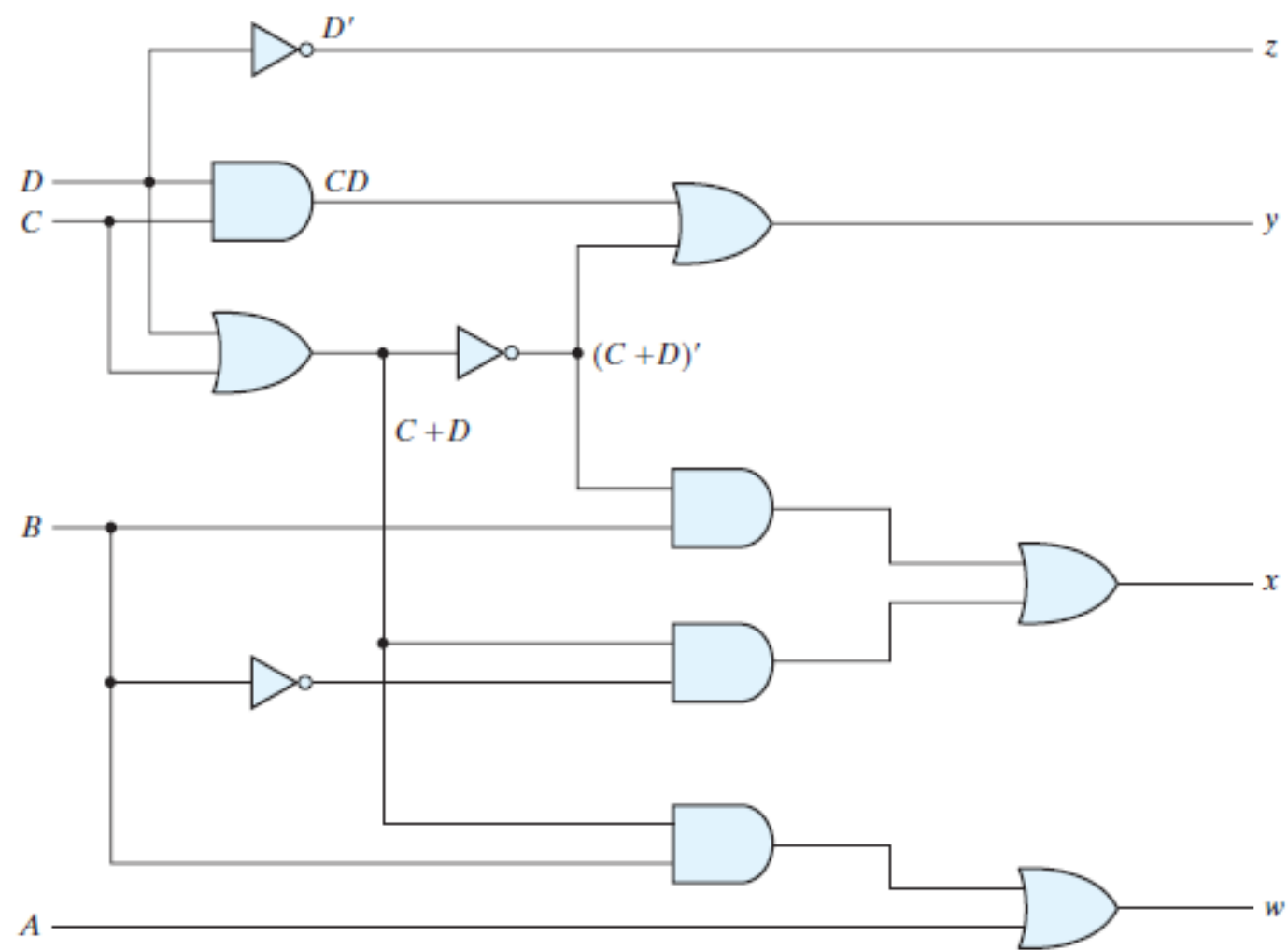


A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	×	×	×	×
1	0	1	1	×	×	×	×
1	1	0	0	×	×	×	×
1	1	0	1	×	×	×	×
1	1	1	0	×	×	×	×
1	1	1	1	×	×	×	×

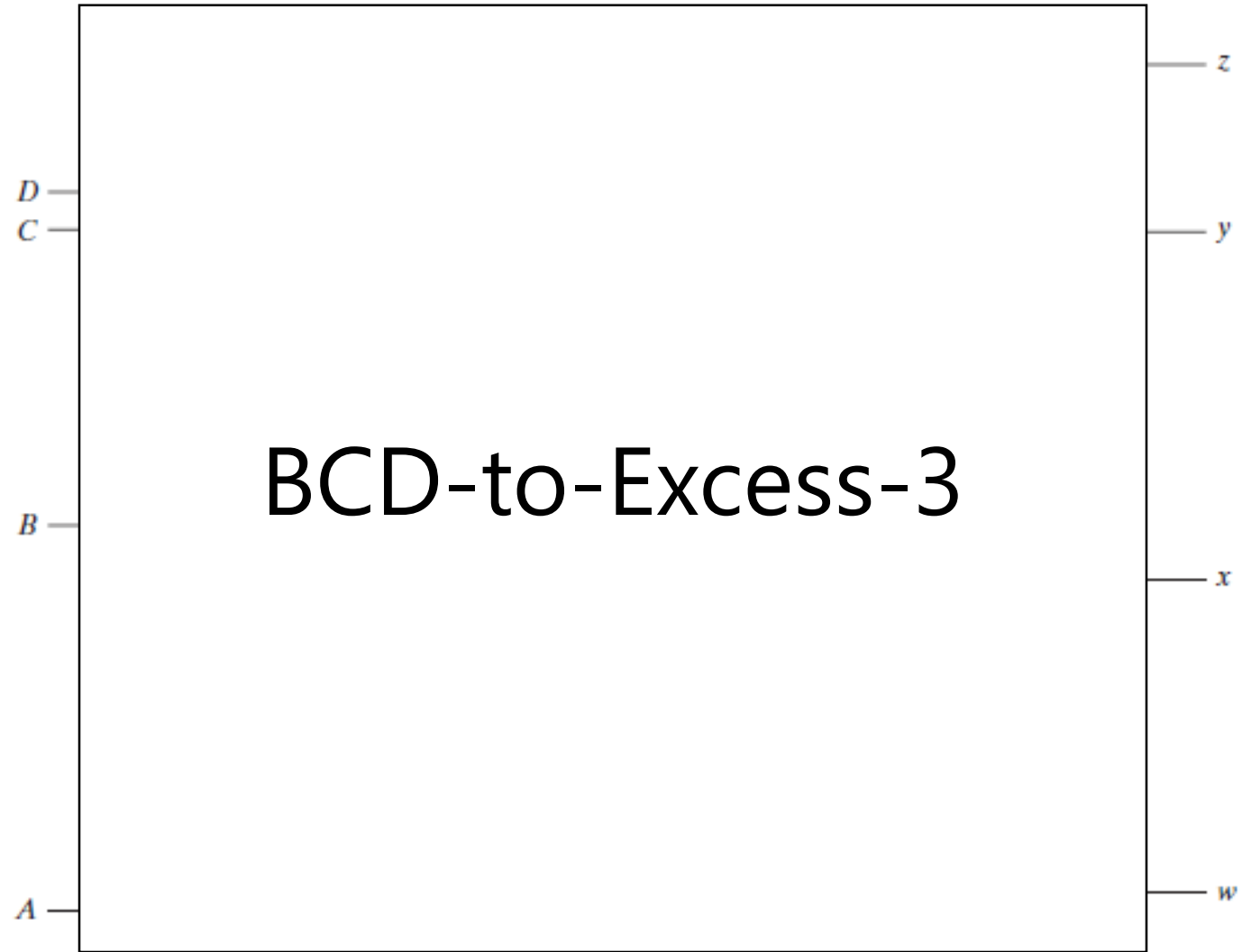
## Chapter 4 Combinational Logic



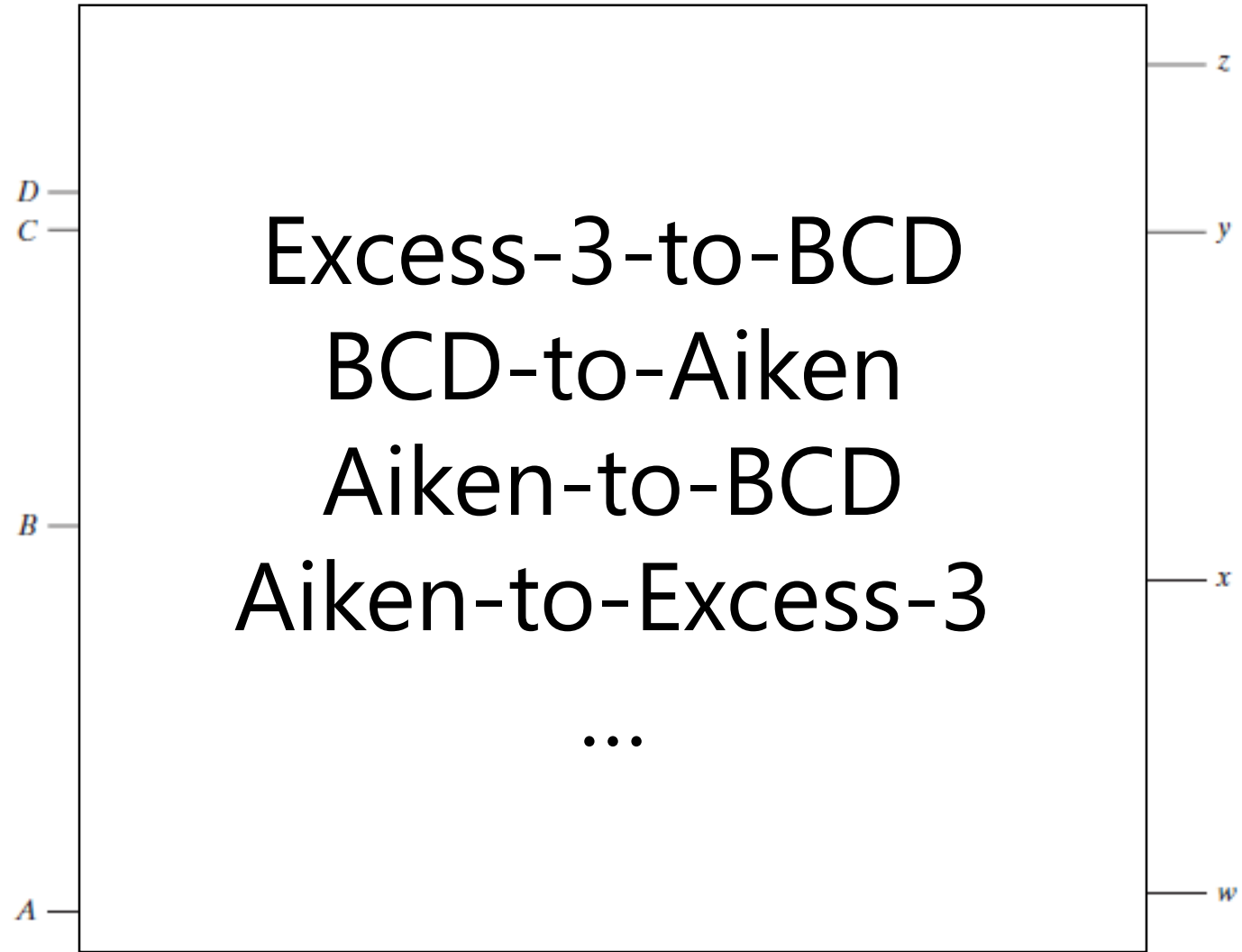
**FIGURE 4.3**  
Maps for BCD-to-excess-3 code converter



**FIGURE 4.4**  
Logic diagram for BCD-to-excess-3 code converter



**FIGURE 4.4**  
Logic diagram for BCD-to-excess-3 code converter



---

# Combinational Logic

## Binary Code Encoder

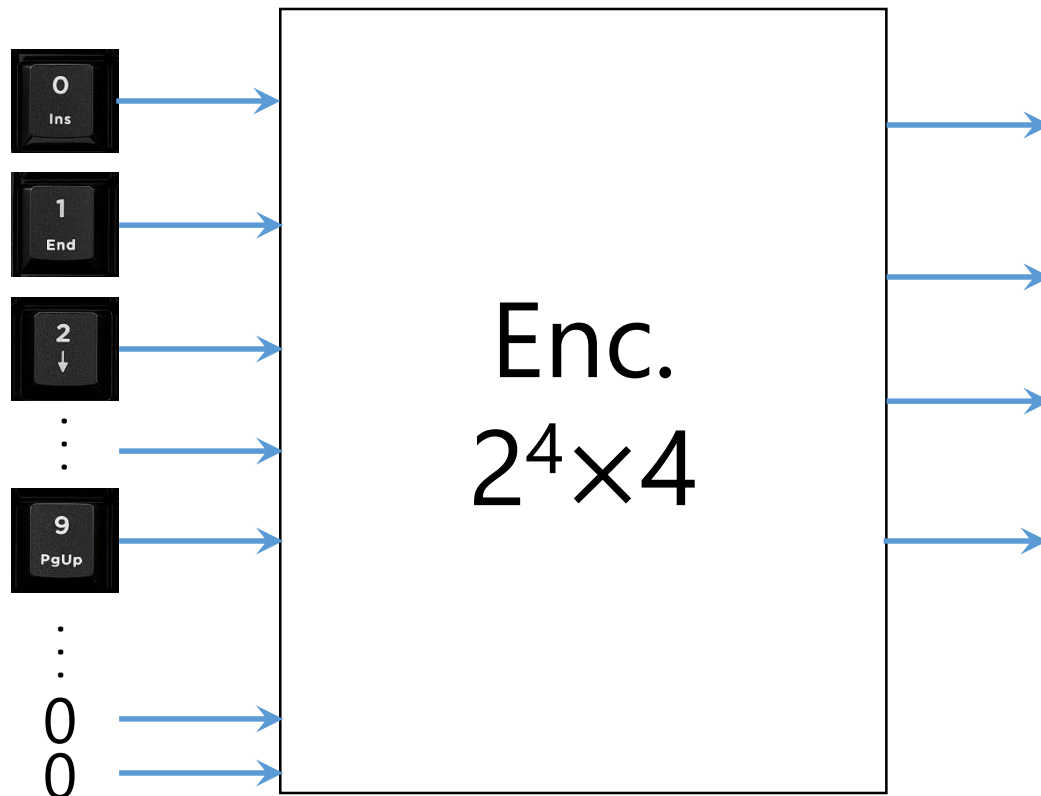
---

---

# Keyboard-to-Binary Code

## Binary Code Encoder

---



Binary Number  
BCD  
Excess-3  
Aiken  
Gray  
...



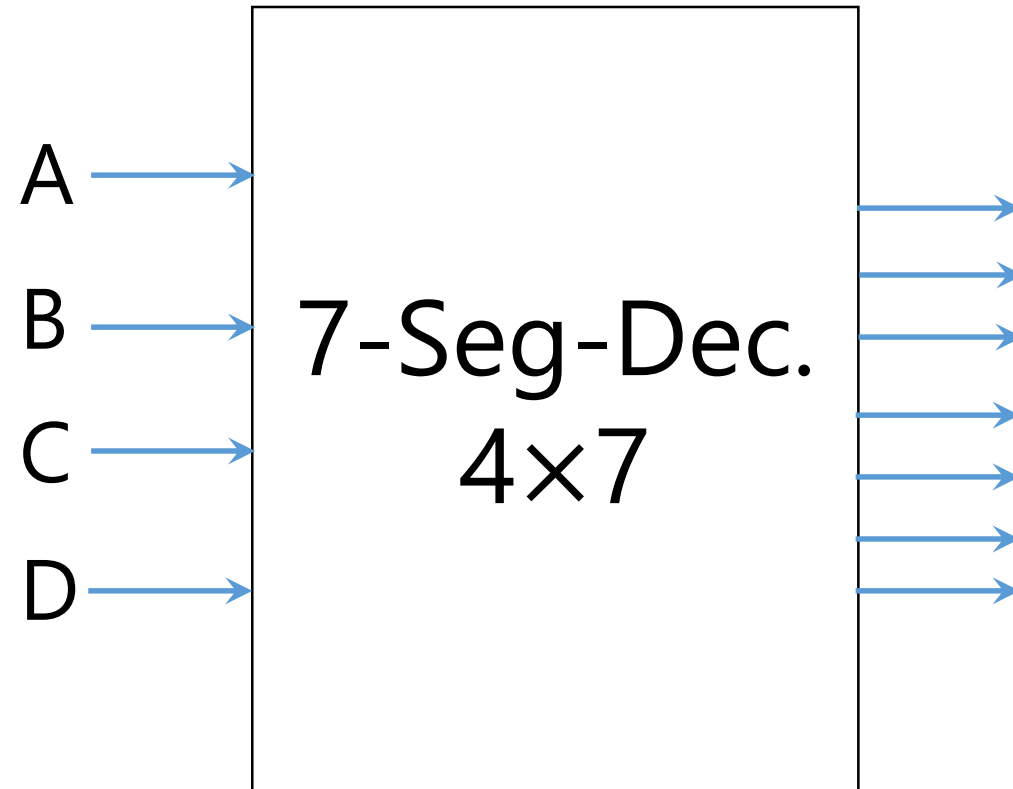
---

# Combinational Logic

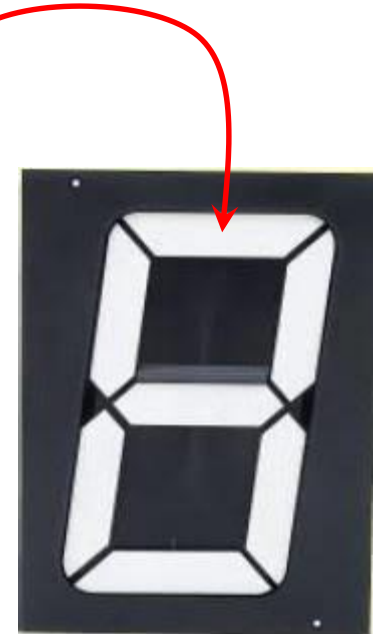
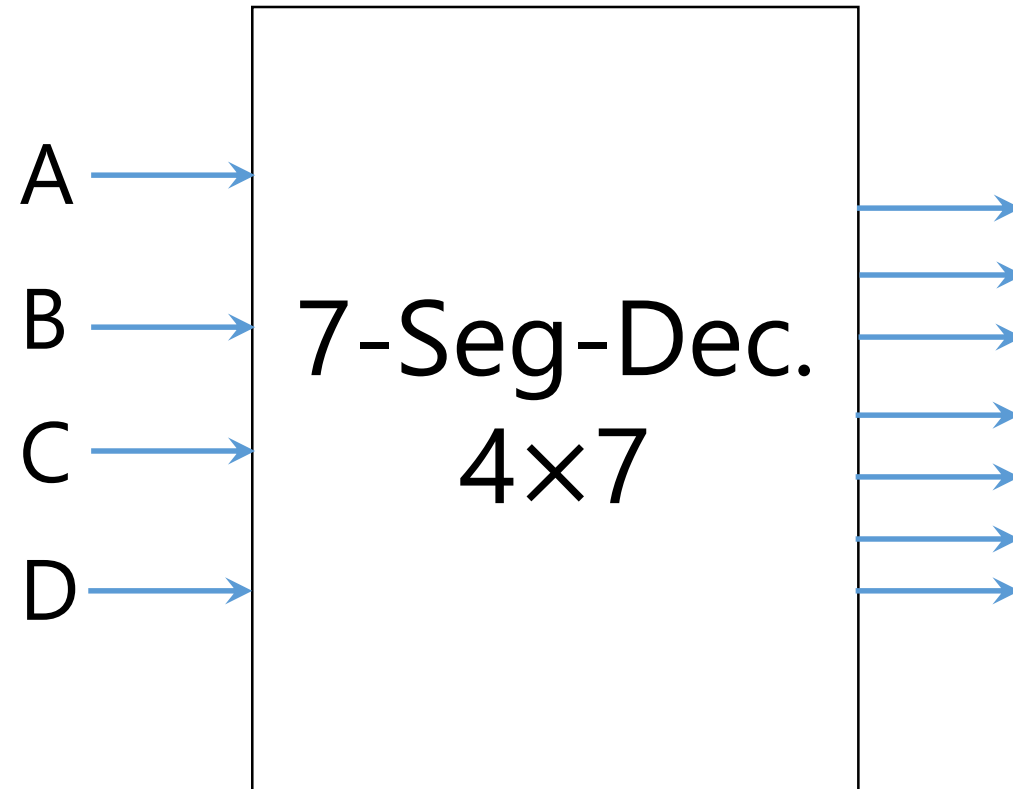
## Display Decoder

---

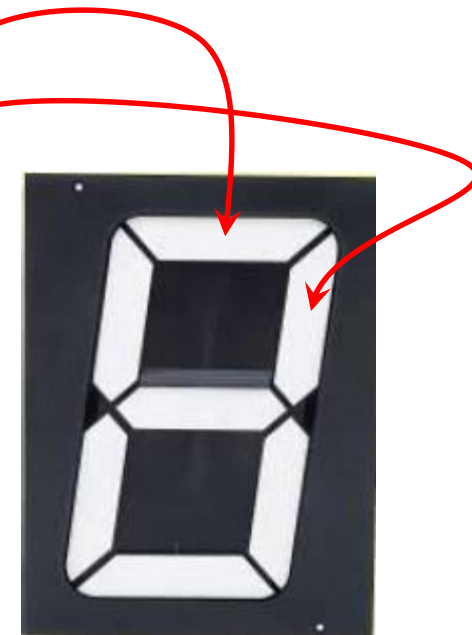
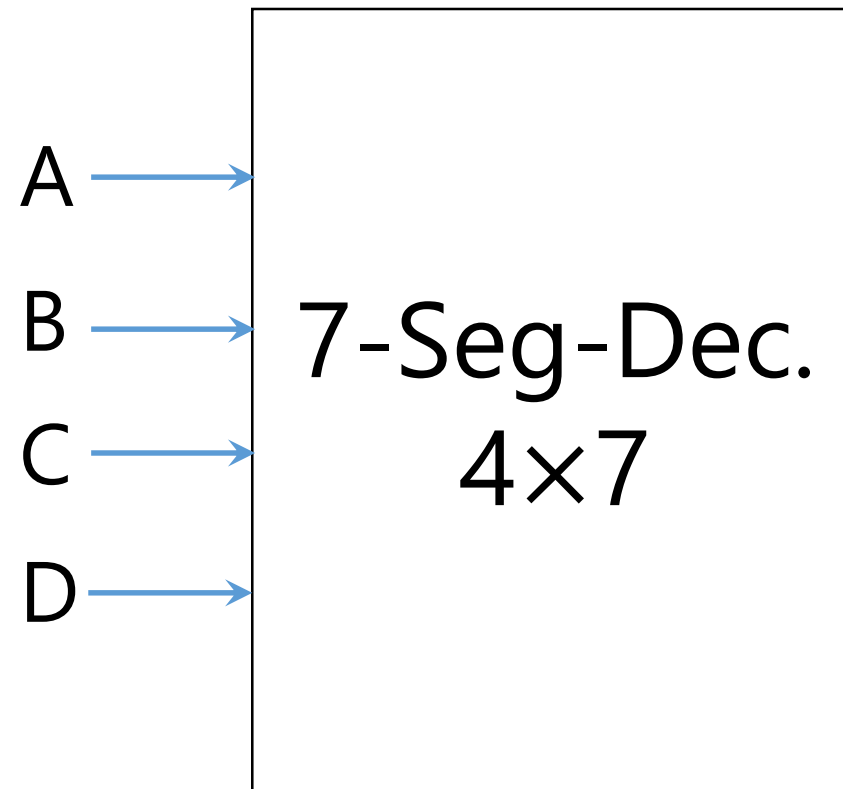
Binary Number  
BCD  
Excess-3  
Aiken  
Gray  
...



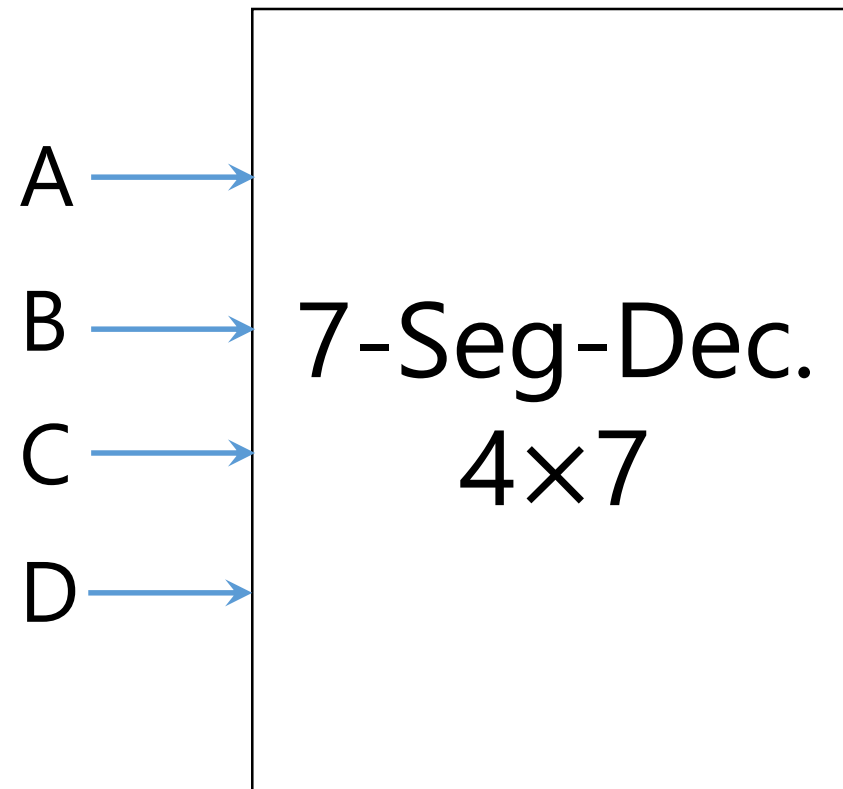
Binary Number  
BCD  
Excess-3  
Aiken  
Gray  
...



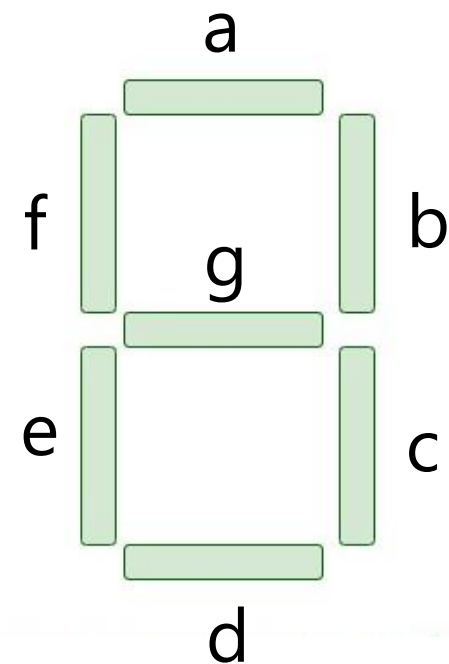
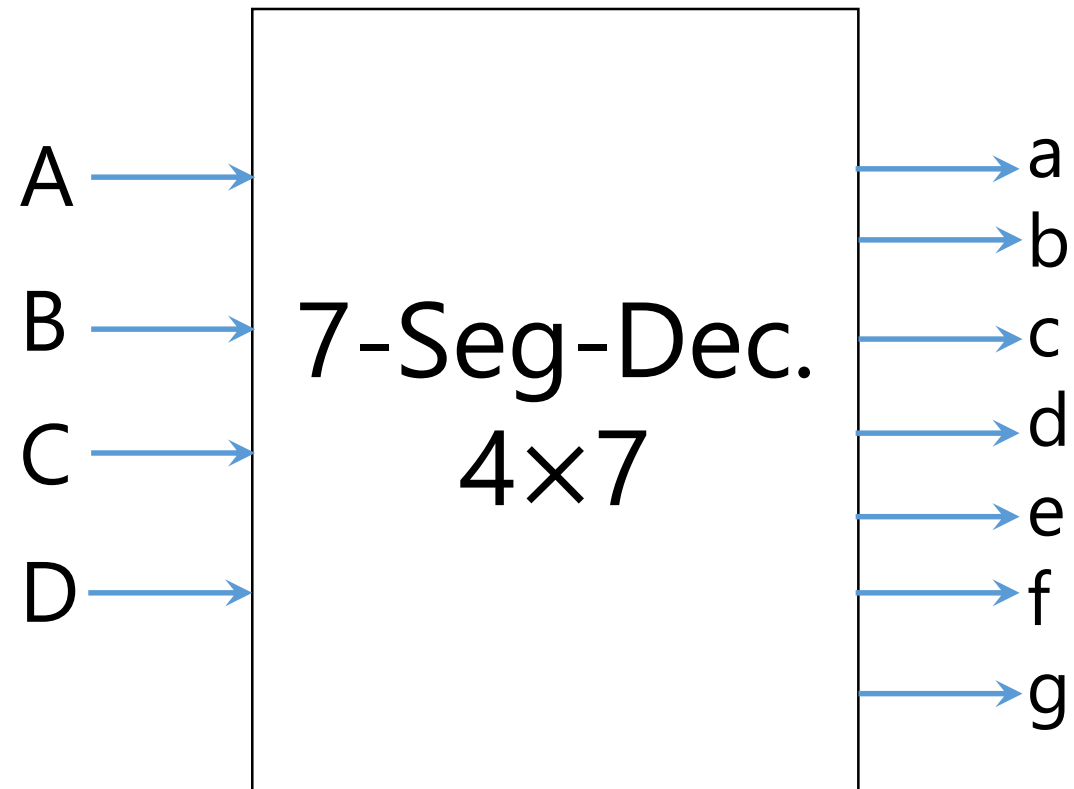
Binary Number  
BCD  
Excess-3  
Aiken  
Gray  
...



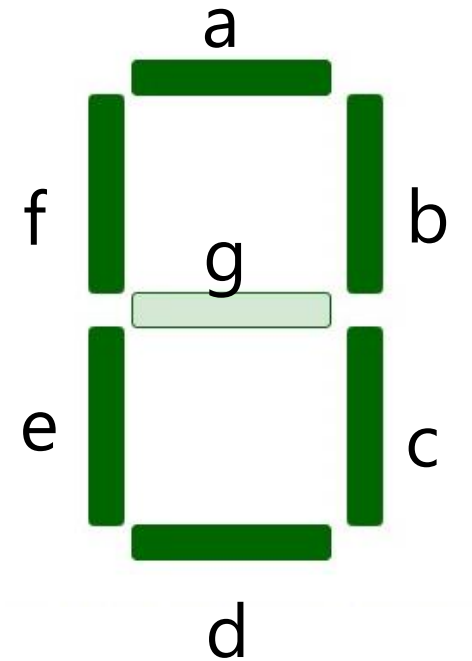
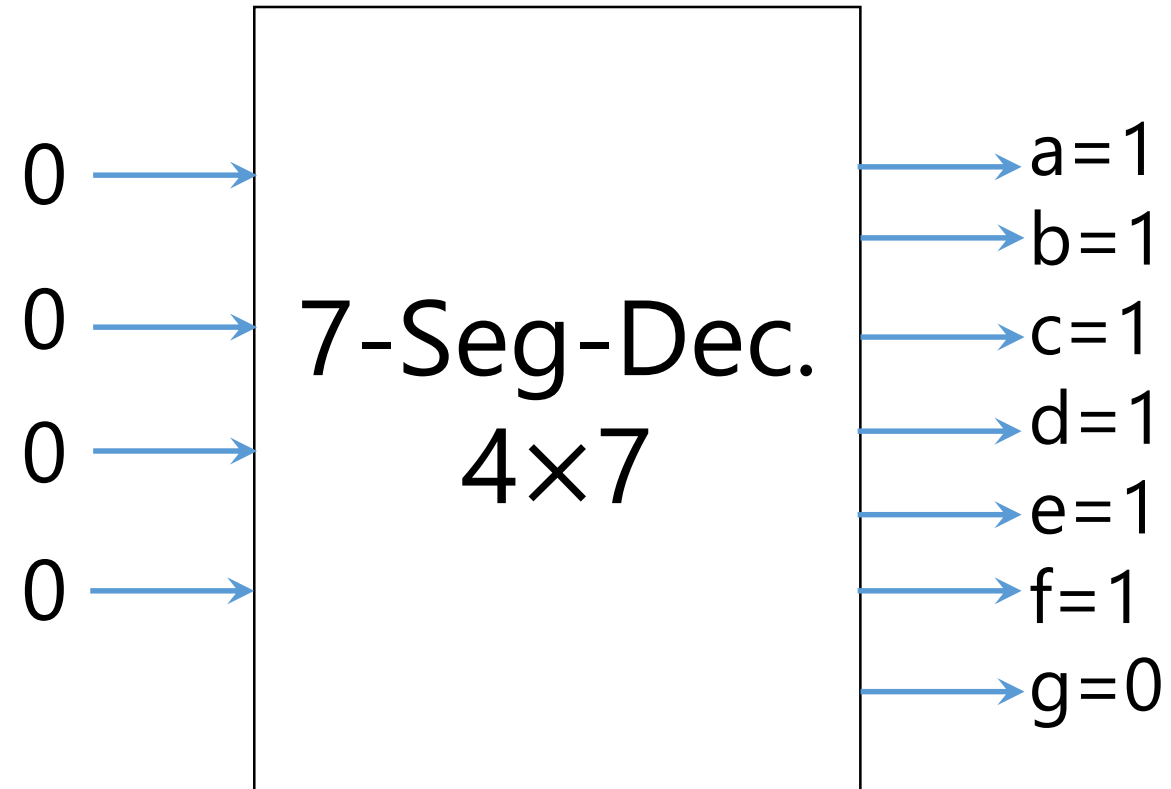
Binary Number  
BCD  
Excess-3  
Aiken  
Gray  
...



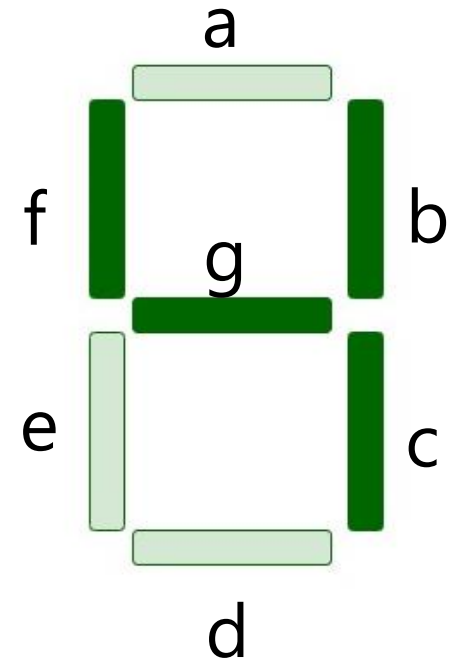
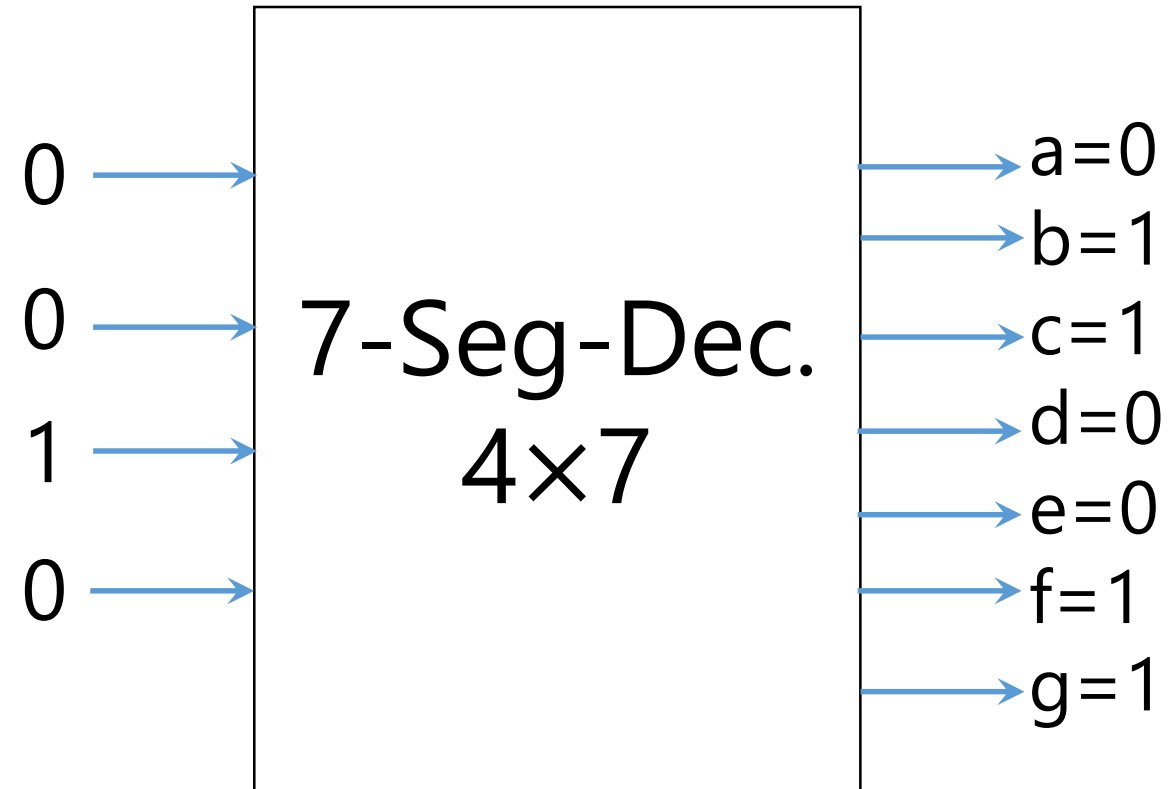
Binary Number  
BCD  
Excess-3  
Aiken  
Gray  
...



Binary Number

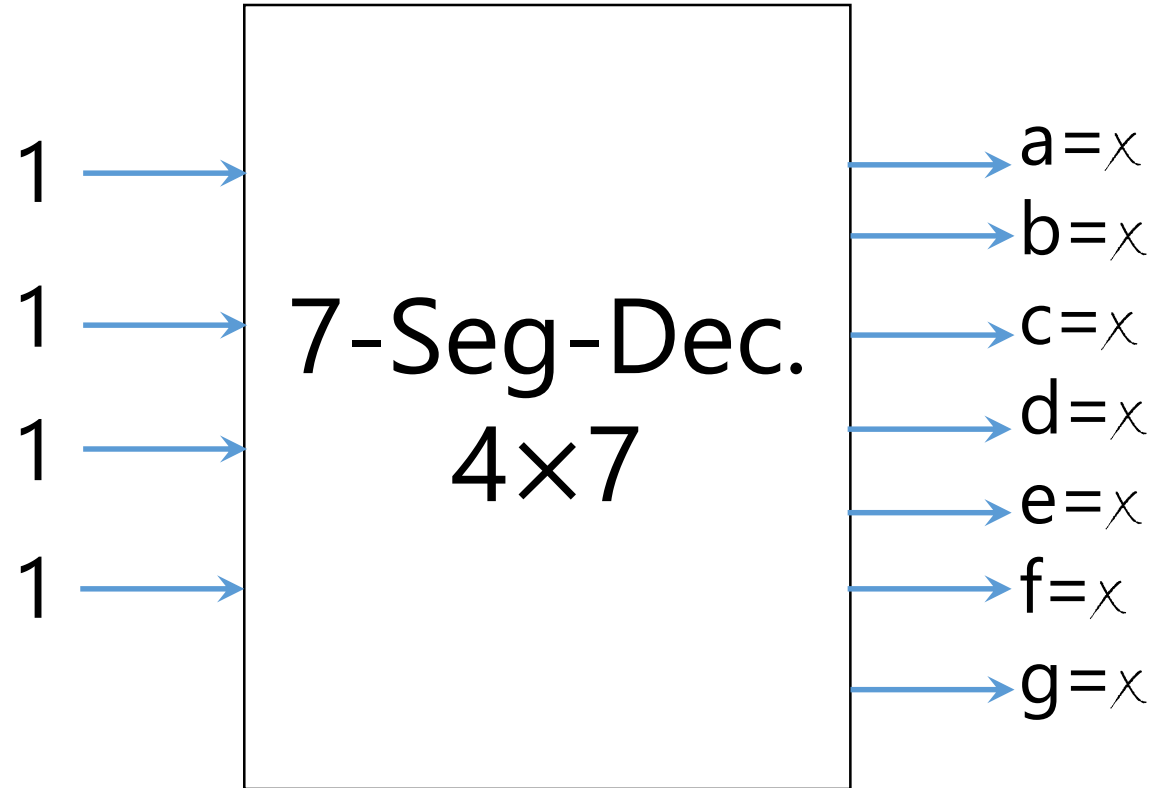


Binary Number



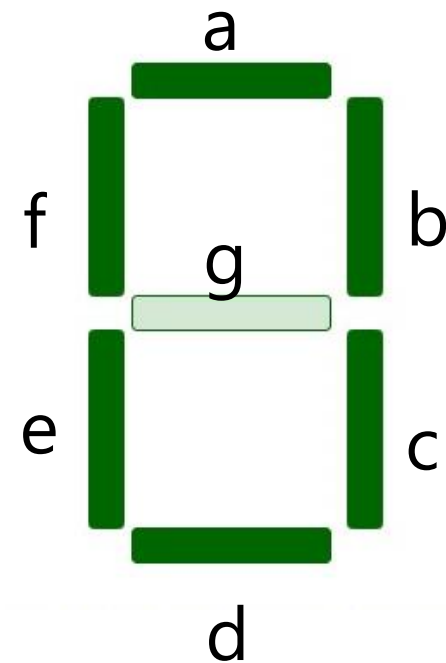
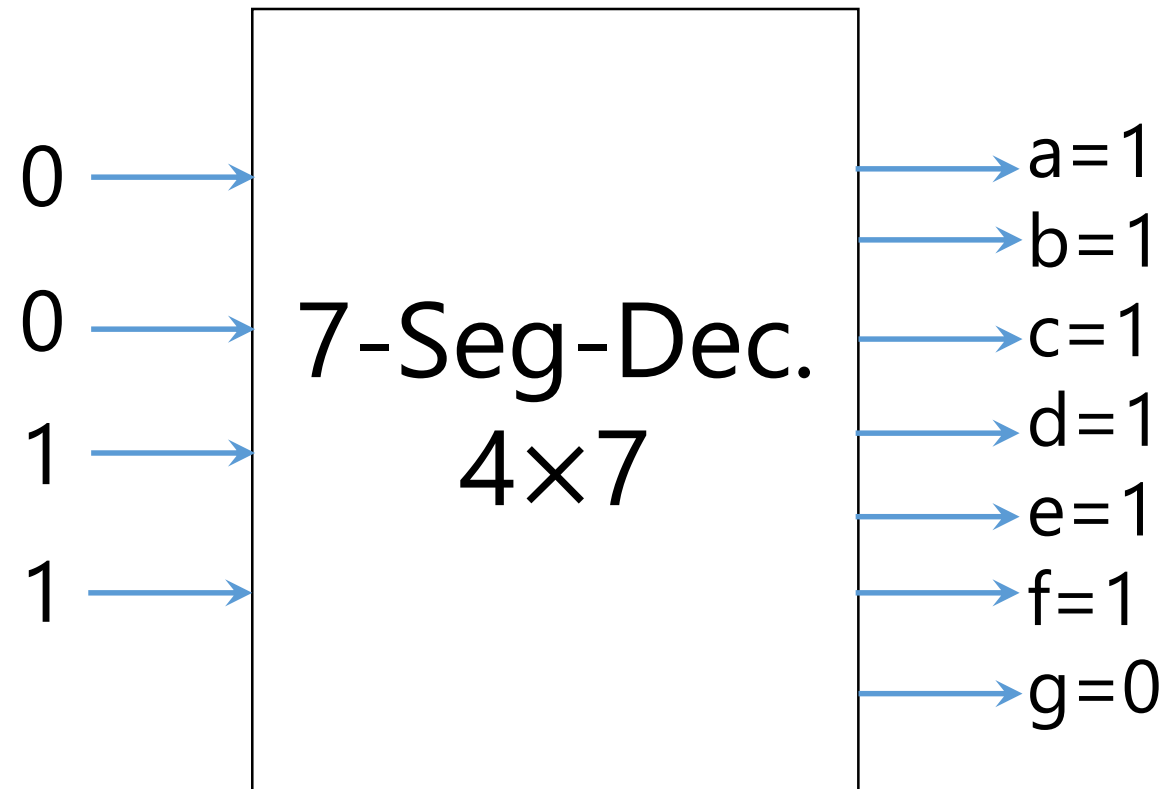


Binary Number

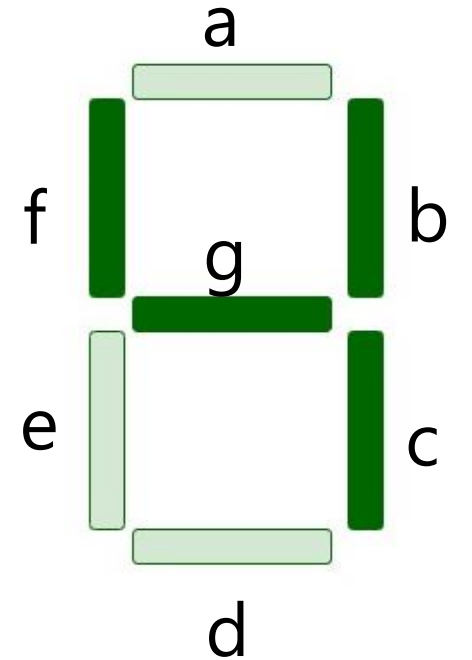
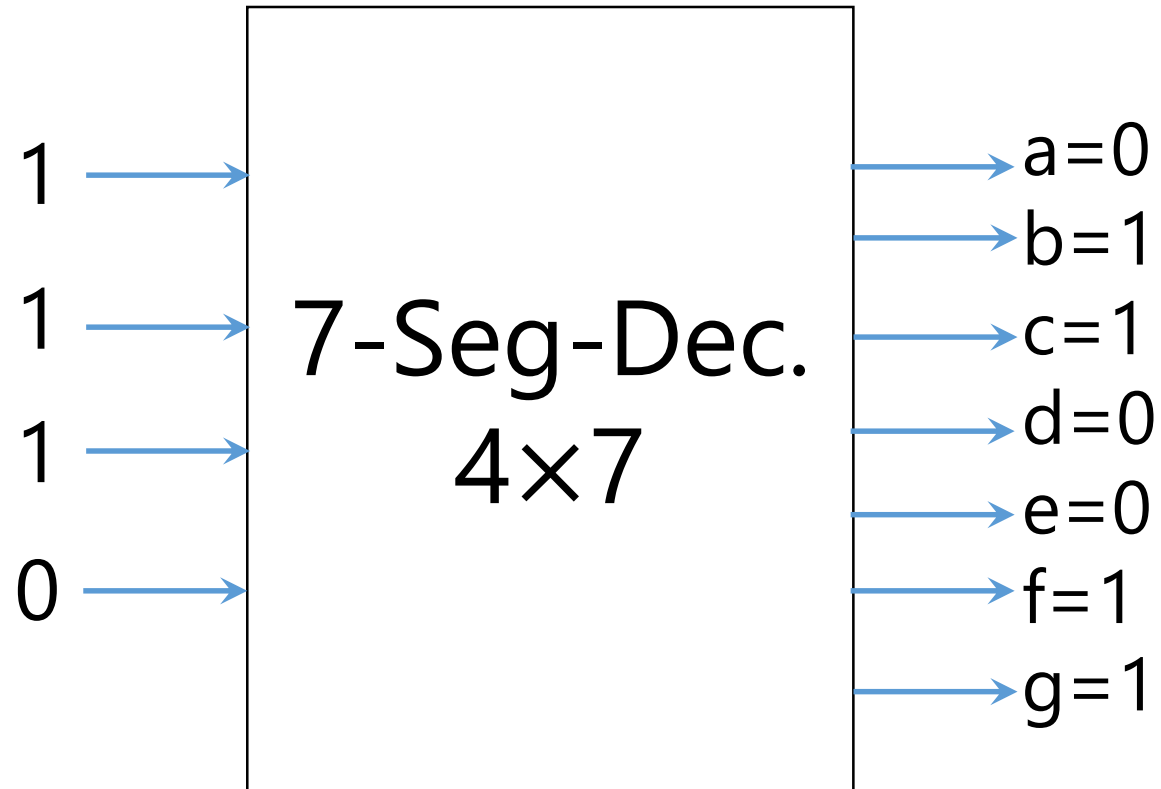


From 10 to 15, don't care conditions!

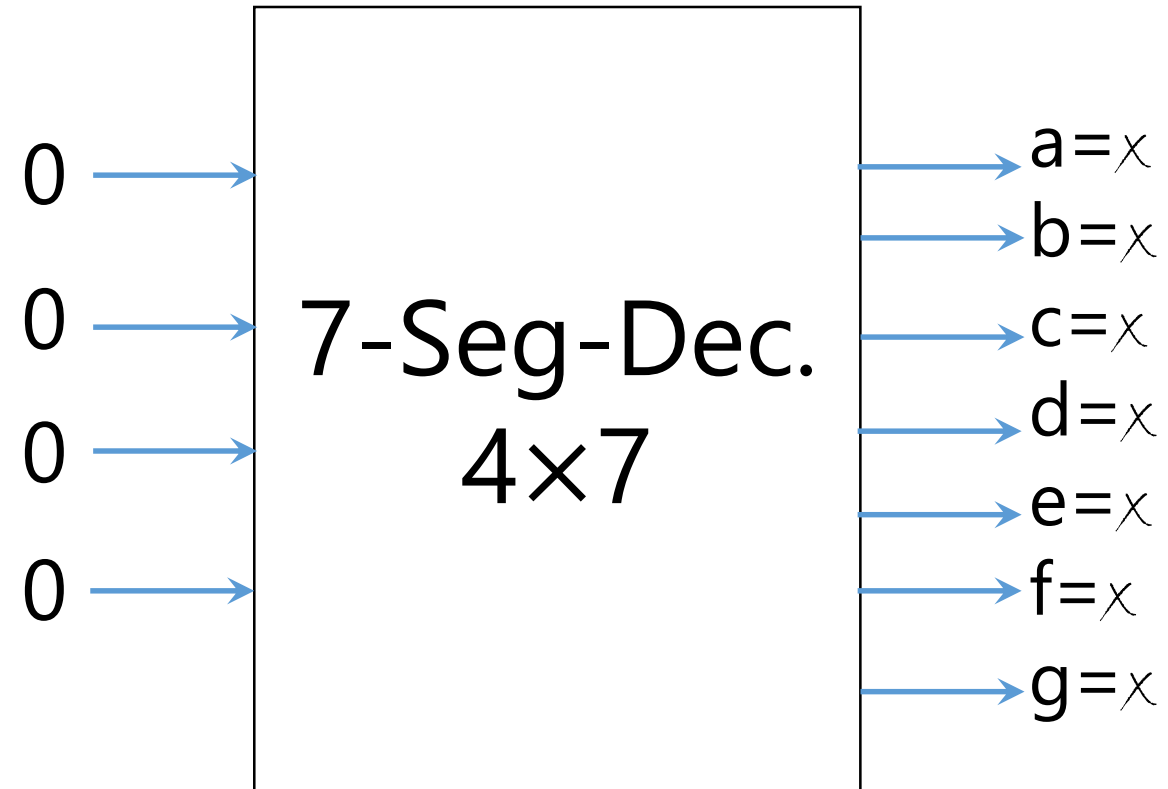
Excess-3



Excess-3

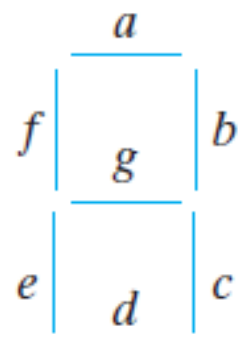


Excess-3



0,1,2,13,14,15, don't care conditions!

- 4.9** An ABCD-to-seven-segment decoder is a combinational circuit that converts a decimal digit in BCD to an appropriate code for the selection of segments in an indicator used to display the decimal digit in a familiar form. The seven outputs of the decoder ( $a, b, c, d, e, f, g$ ) select the corresponding segments in the display, as shown in Fig. P4.9(a). The numeric display chosen to represent the decimal digit is shown in Fig. P4.9(b). Using a truth table and Karnaugh maps, design the BCD-to-seven-segment decoder using a minimum number of gates. The six invalid combinations should result in a blank display. (HDL—see Problem 4.51.)



(a) Segment designation



(b) Numerical designation for display

**FIGURE P4.9**

---

# Combinational Logic

## Binary Code Arithmetic

---

---

# Combinational Logic

## BCD Adder

---

Book: Page 144-146