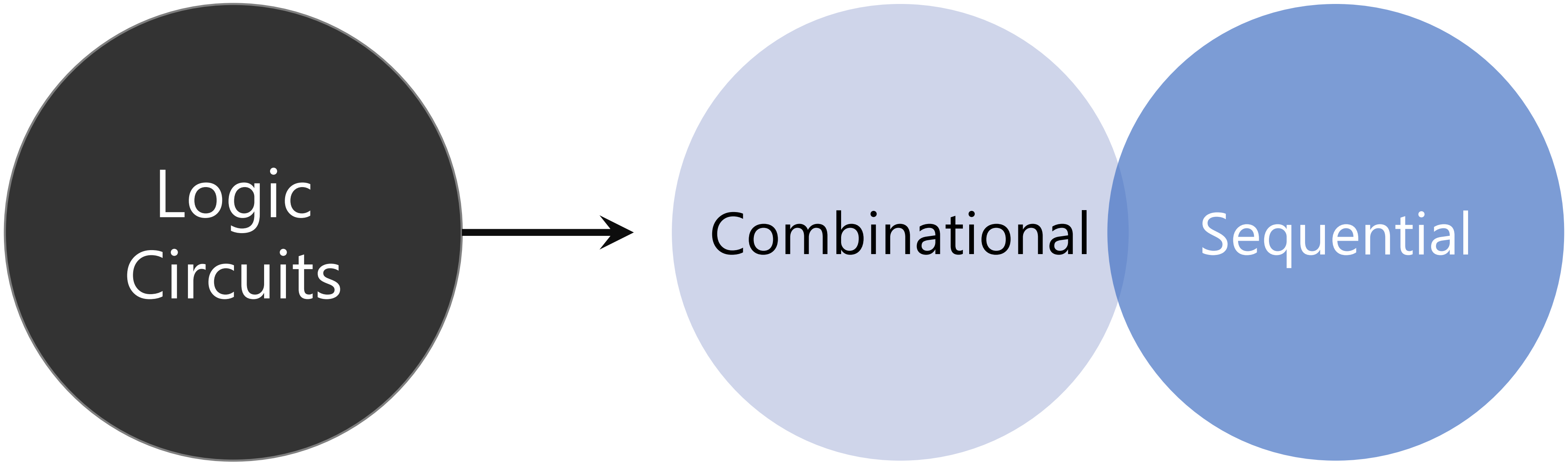

Logic
Circuits
Combinational
Sequential

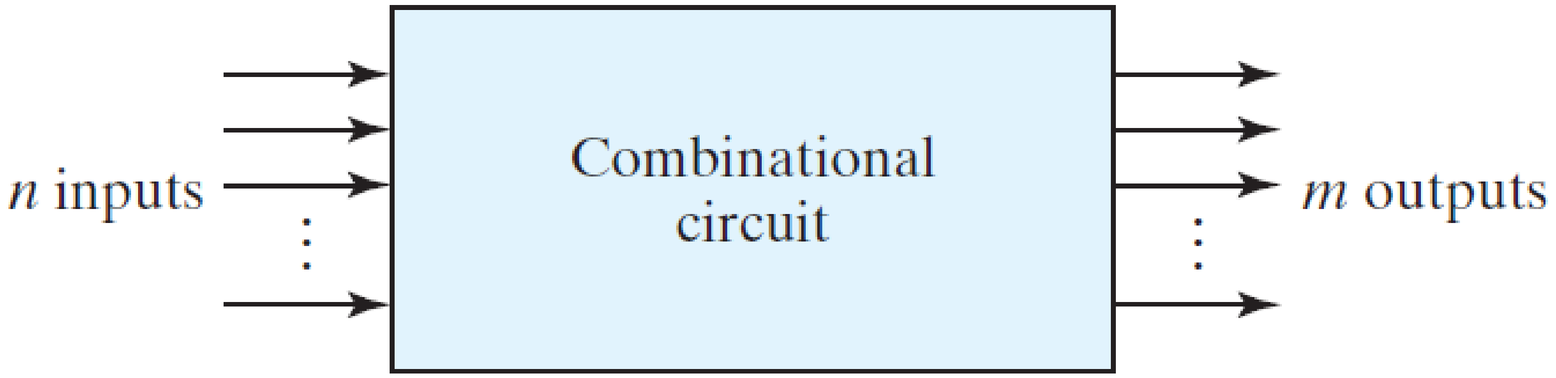


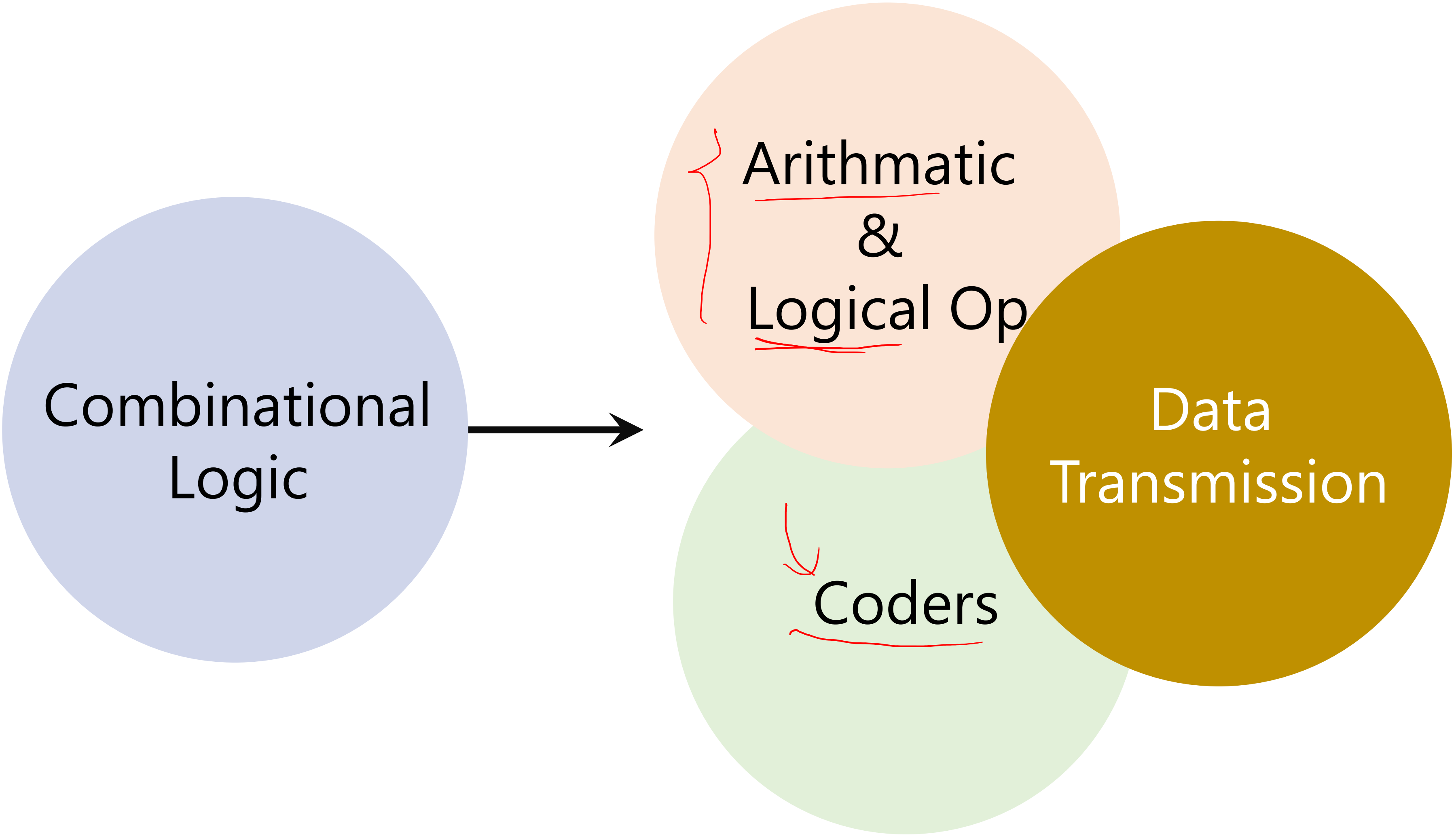
FIGURE 4.1
Block diagram of combinational circuit

Combinational Logic

aka. Combinational Circuit

Combination of logic gates on the present inputs → the outputs *at any time*!

A combinational circuit performs an operation that can be specified logically by a set of Boolean functions.


Combinational
Logic
Arithmatic
&
Logical Op
Coders
Data
Transmission

THE INTERNATIONAL
Calculator Collector

Spring 1993 — Issue No. 1

The Cal Tech circa 1967 — *Photo Courtesy Texas Instruments*

The Beginning

If you're past your mid-30s, you probably remember your first simple hand-held calculator costing over $50 (in early 1970's dollars). Depending how much older you are, your first could have been upwards to $400. And we're just talking the basic four functions here — addition, subtraction, multiplication, and division. Percentage and memory features were extra (if they were even available at that point in time).

Company Profile:

Bowmar

Who can forget the "Bowmar Brain" series of calculators from the early '70s?

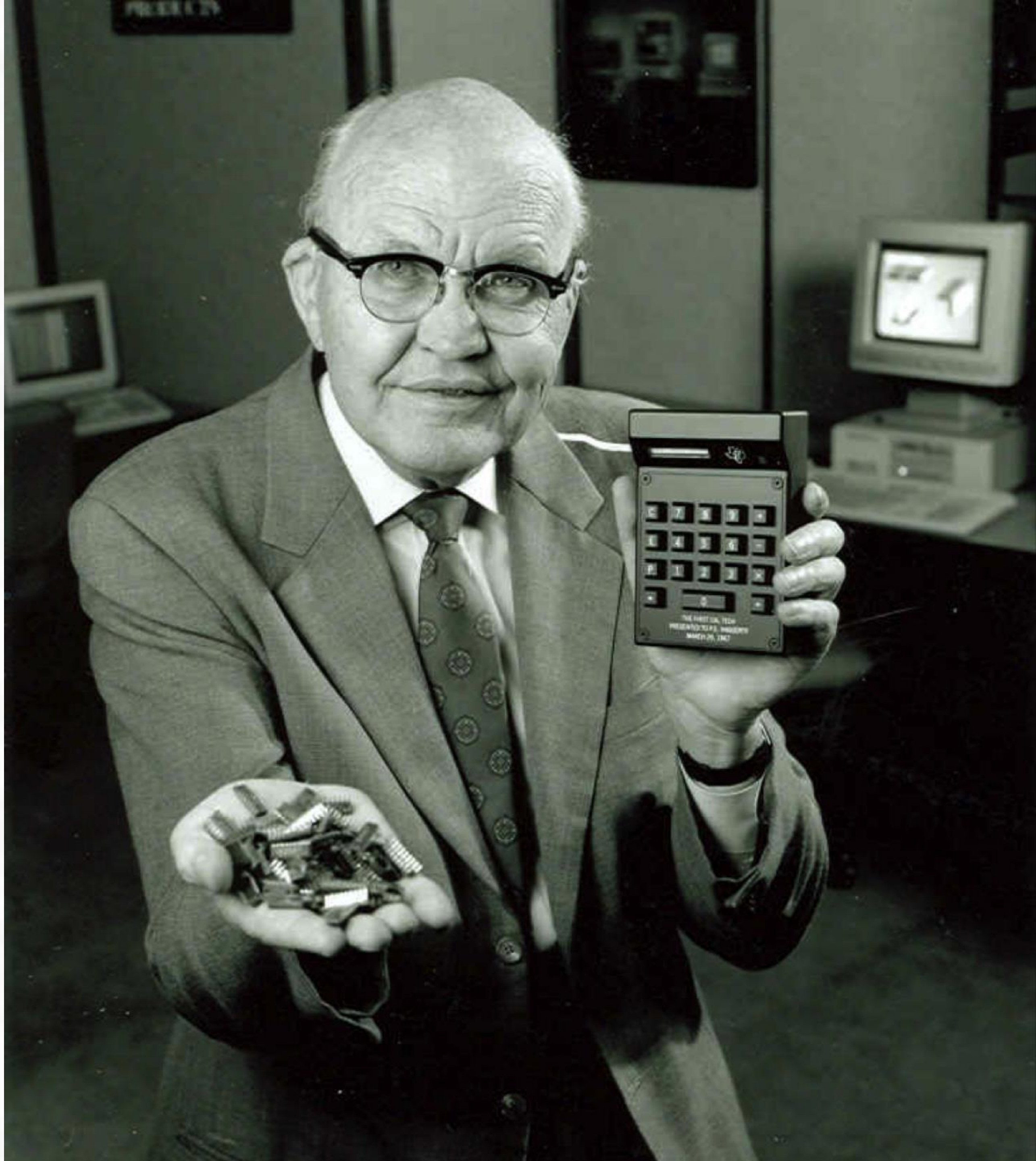
Bowmar was the first American company that made and sold their own line of portable electronic machines.

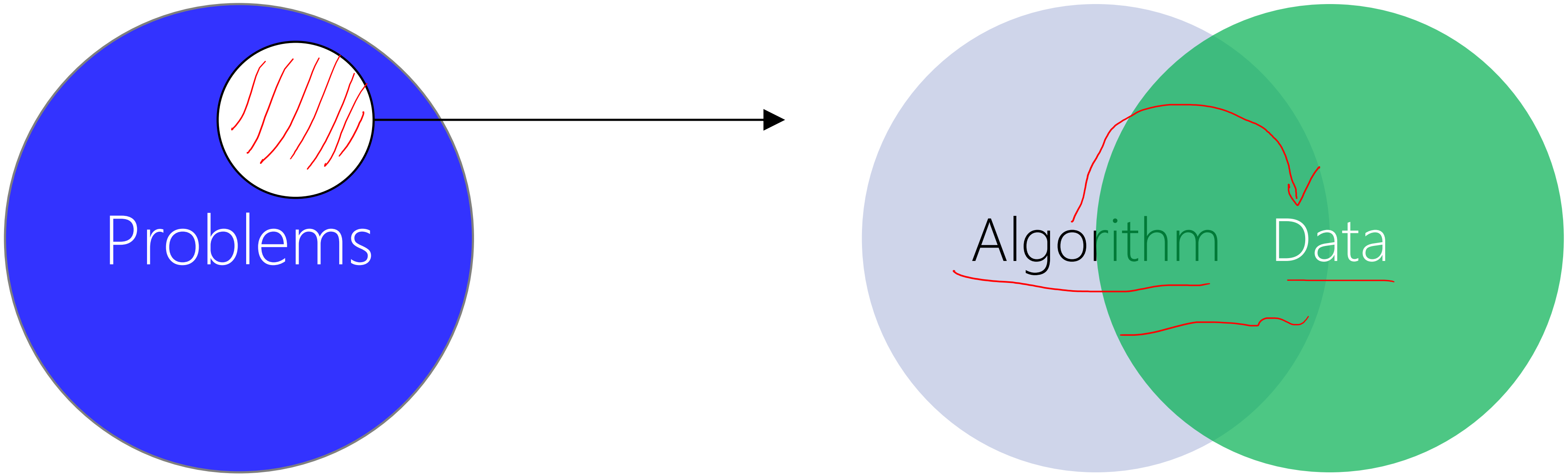
The story starts around 1970 when Bowmar, then a manufacturer of Light Emitting Diodes (LEDs), tried to sell their numeric display product to Japanese manufacturers for use in their electronic products.

Bowmar wasn't too successful. The Japanese were using a flourescent style display that was cheaper and had a few design features the manufacturers liked better.

So, president Ed White, a consummate entrepreneur, and his staff came up with an even better idea — make the whole electronic calculator themselves.

Up to now, most of the so-called "portable" calculators



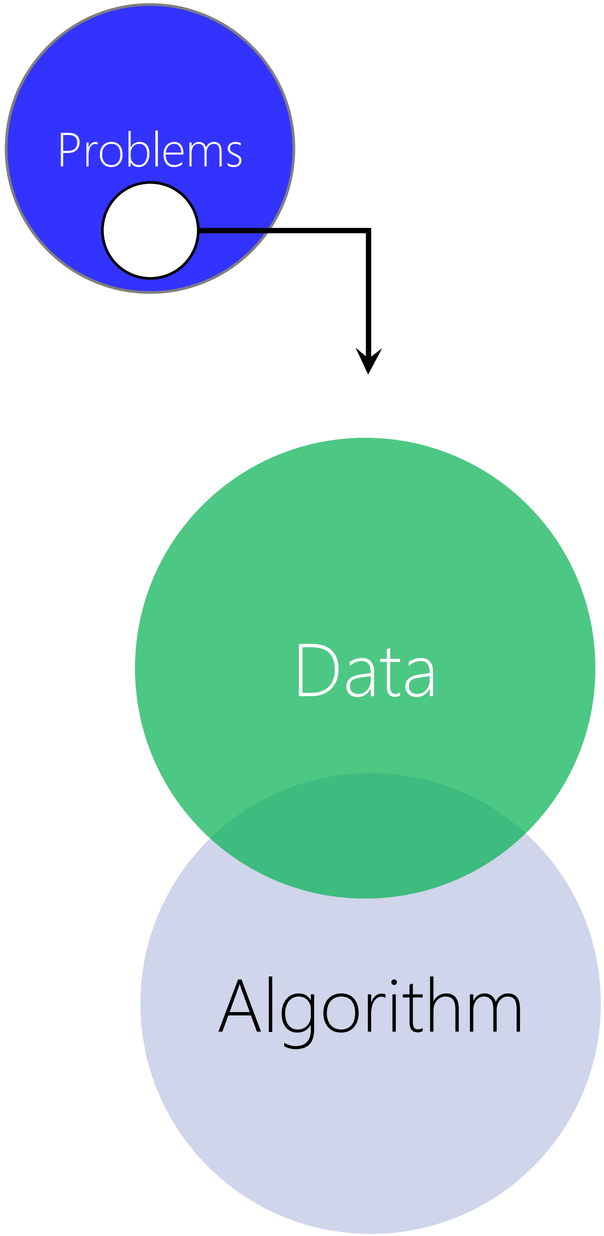

Problems
Algorithm
Data

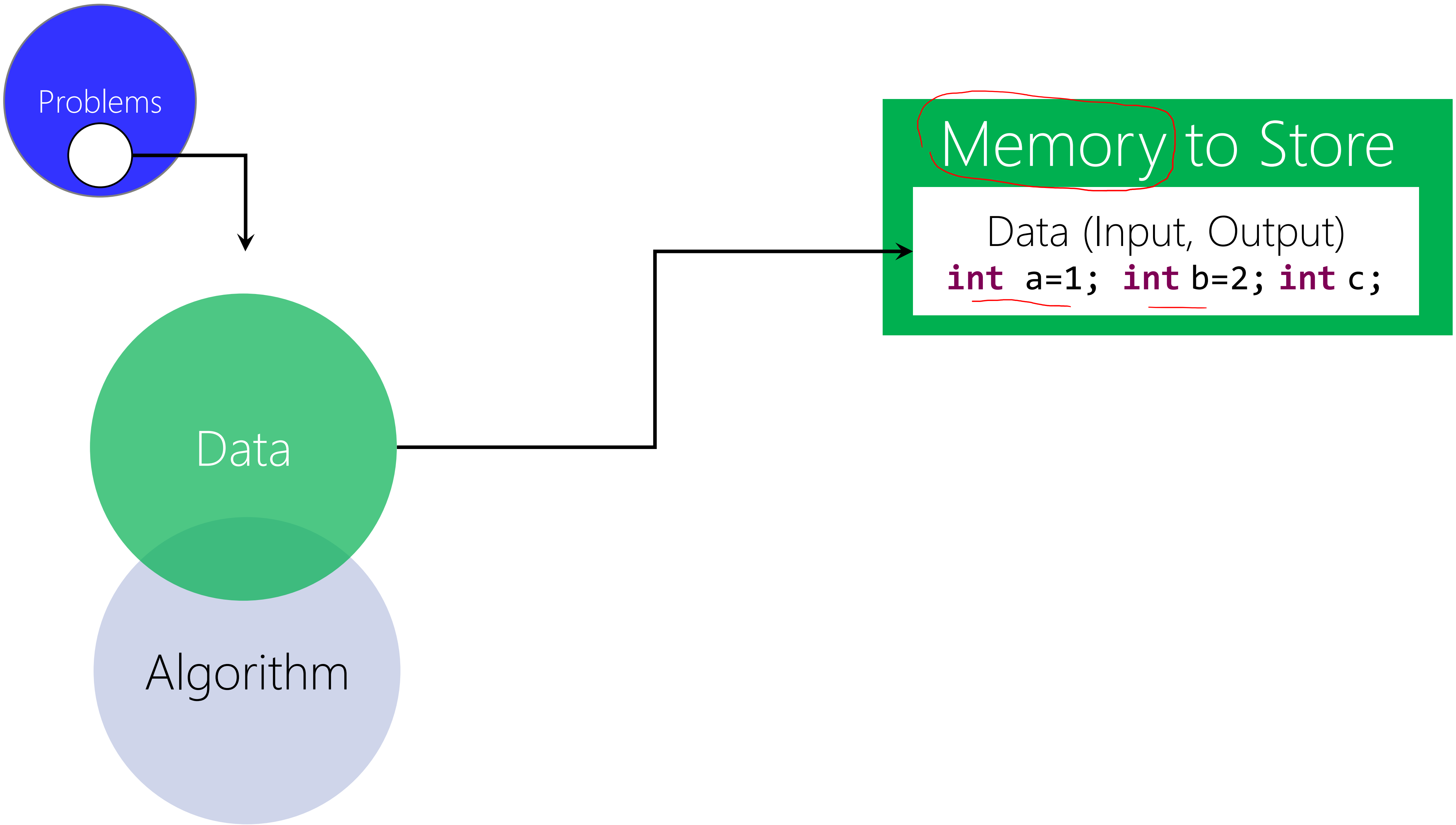
Design a Computer System

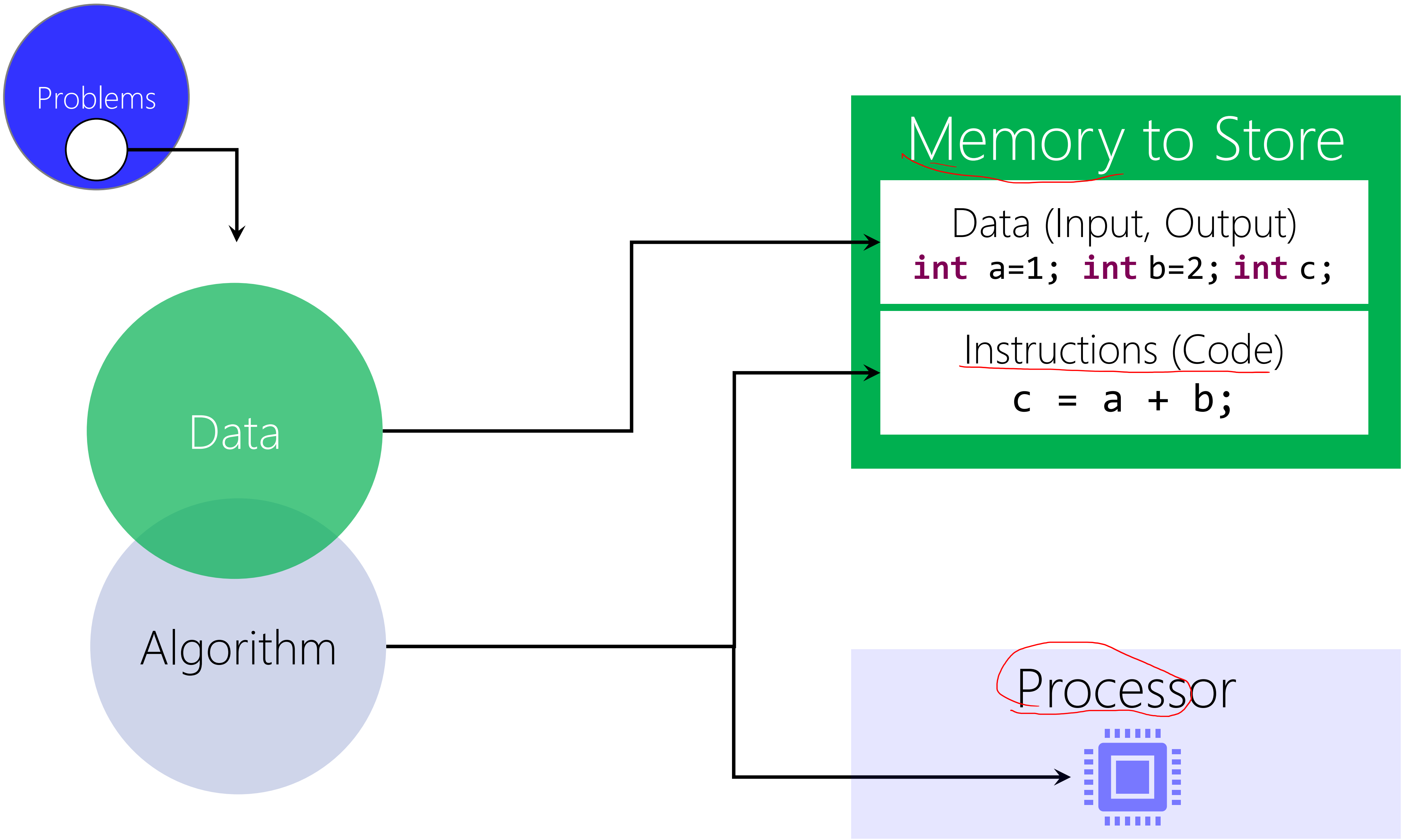
John von Neumann

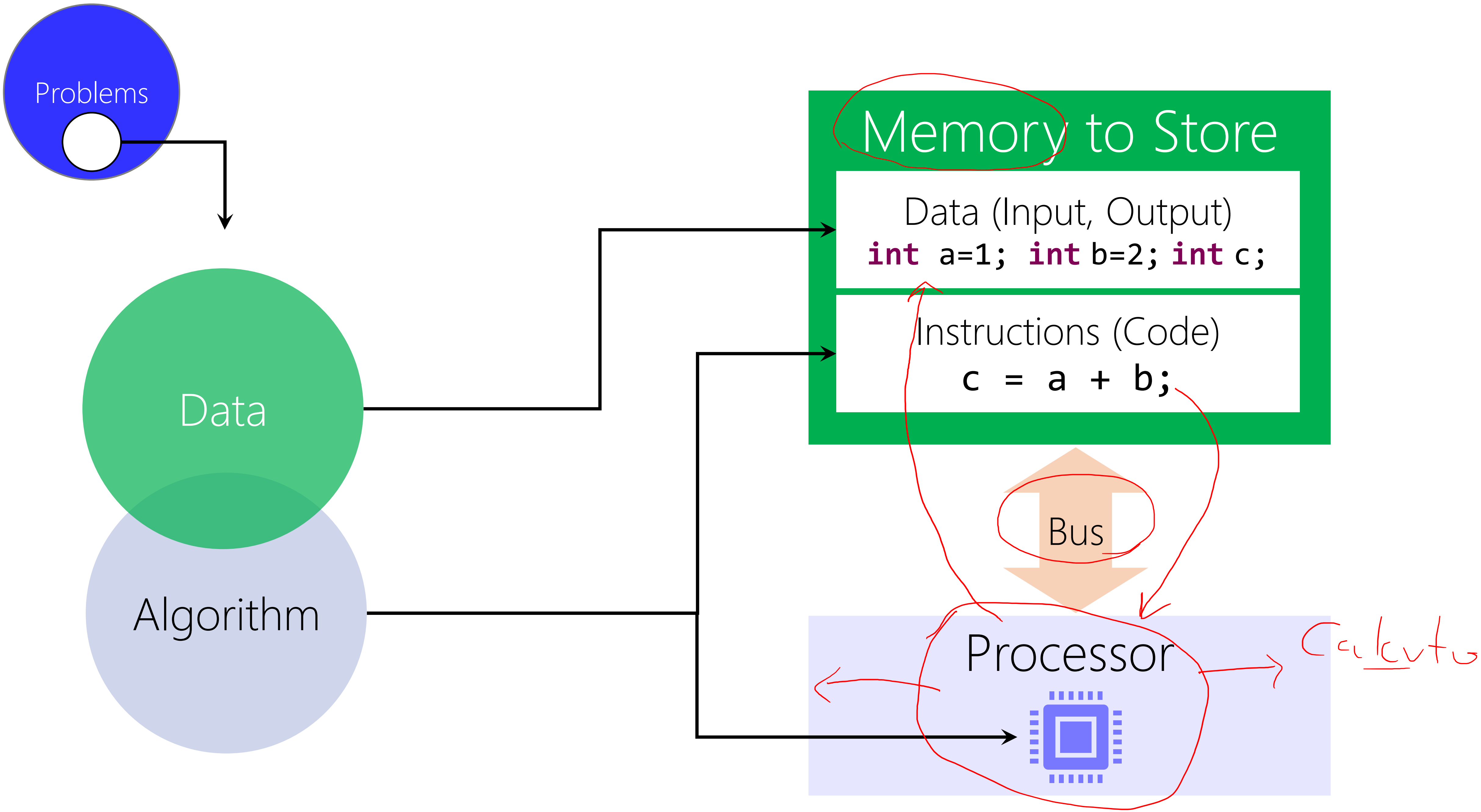
([/vɒn ˈnɔɪmən/](#))
1903 –1957
Mathematician, Physicist, Computer Scientist, Engineer

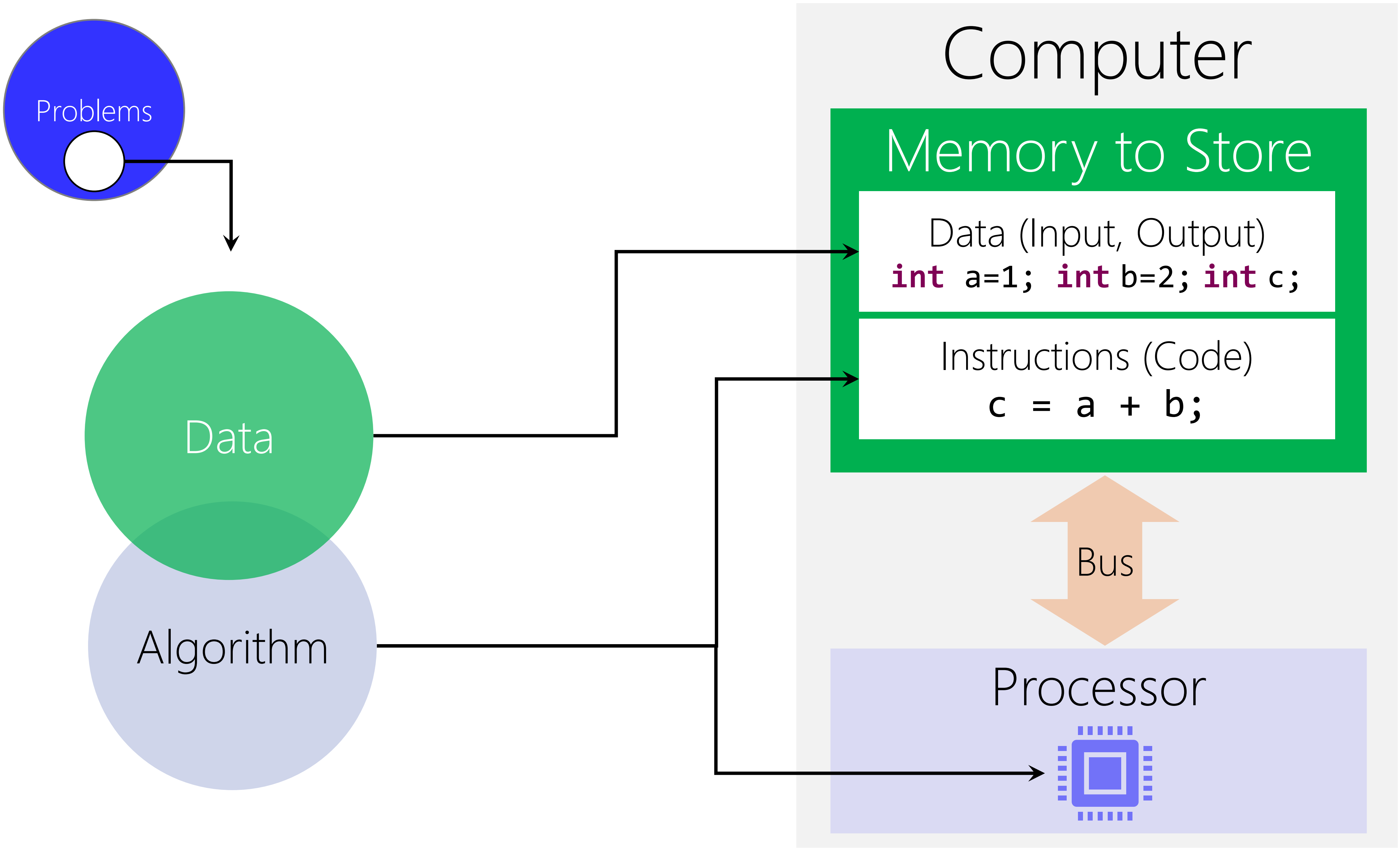
Polymath
He integrated pure and applied sciences. He made major contributions to many fields, including:
- Mathematics
- Physics
- Economics (game theory)
- Computing
- Statistics


Problems
Data
Algorithm


Problems
Data
Algorithm
Memory to Store
Data (Input, Output)
int a=1; int b=2; int c;


Problems
Data
Algorithm
Memory to Store
Data (Input, Output)
int a=1; int b=2; int c;
Instructions (Code)
c = a + b;
Processor

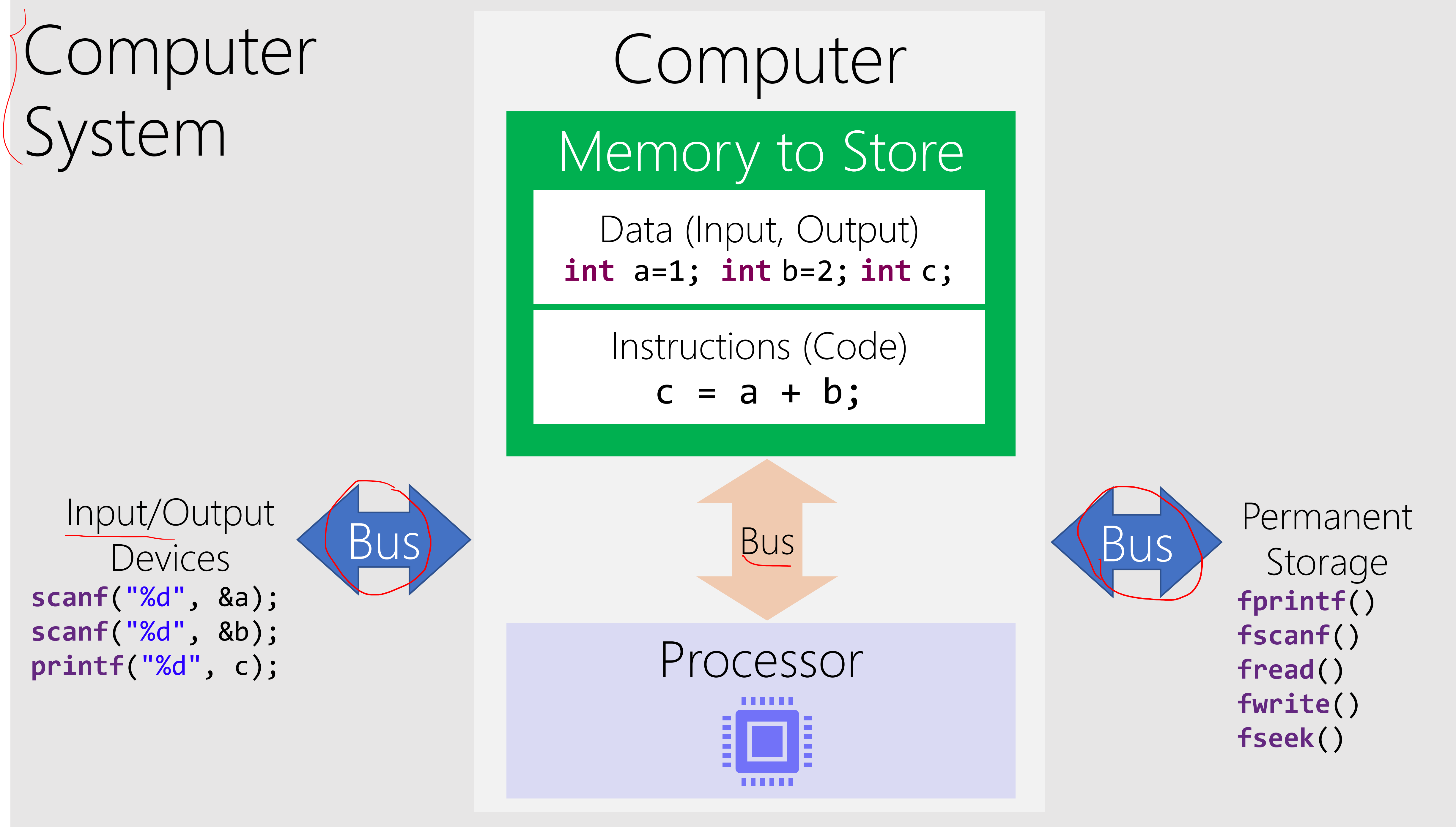

Problems
Data
Algorithm
Memory to Store
Data (Input, Output)
int a=1; int b=2; int c;
Instructions (Code)
c = a + b;
Bus
Processor

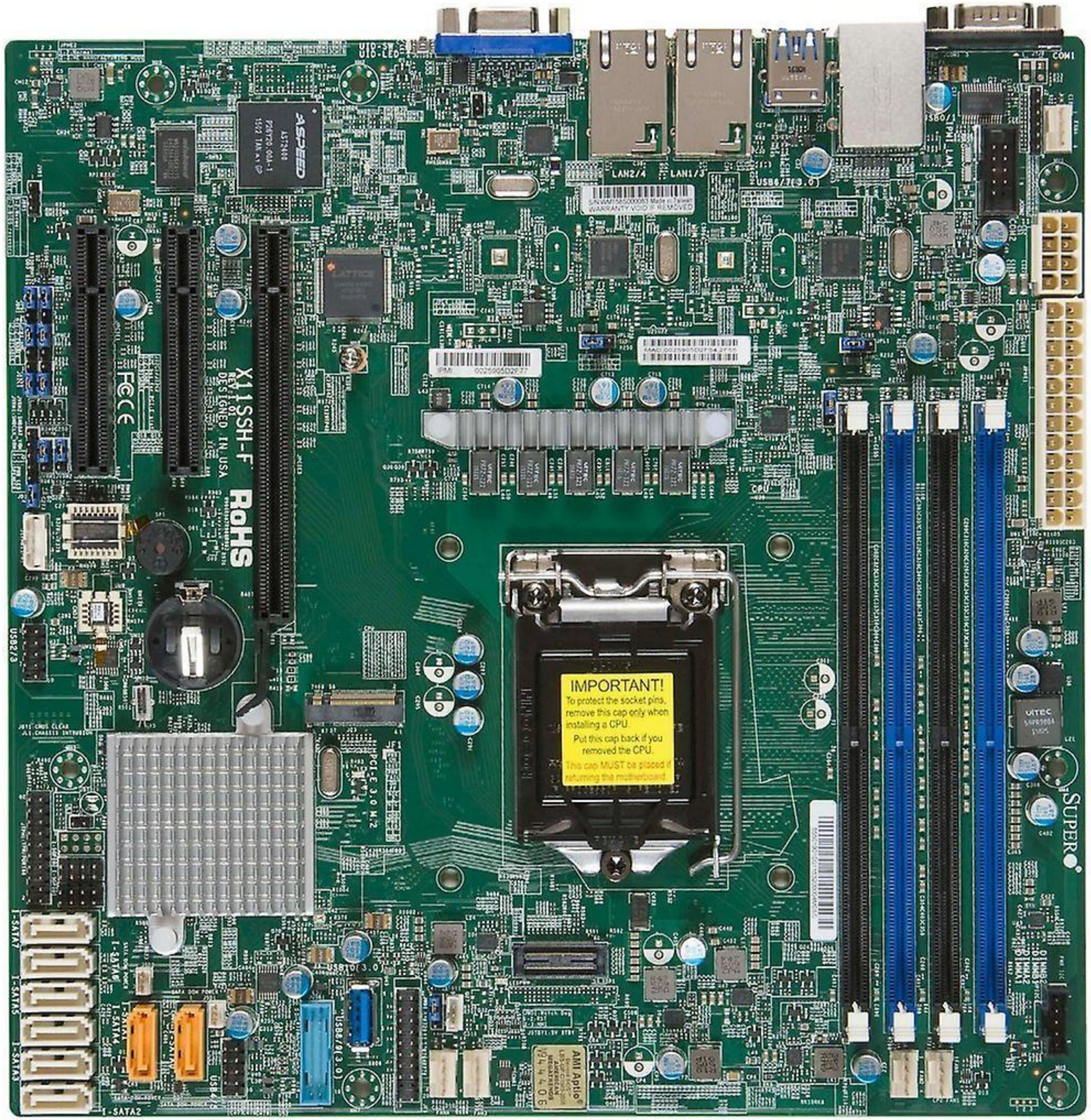

Problems
Data
Algorithm
Computer
Memory to Store
Data (Input, Output)
int a=1; int b=2; int c;
Instructions (Code)
c = a + b;
Bus
Processor

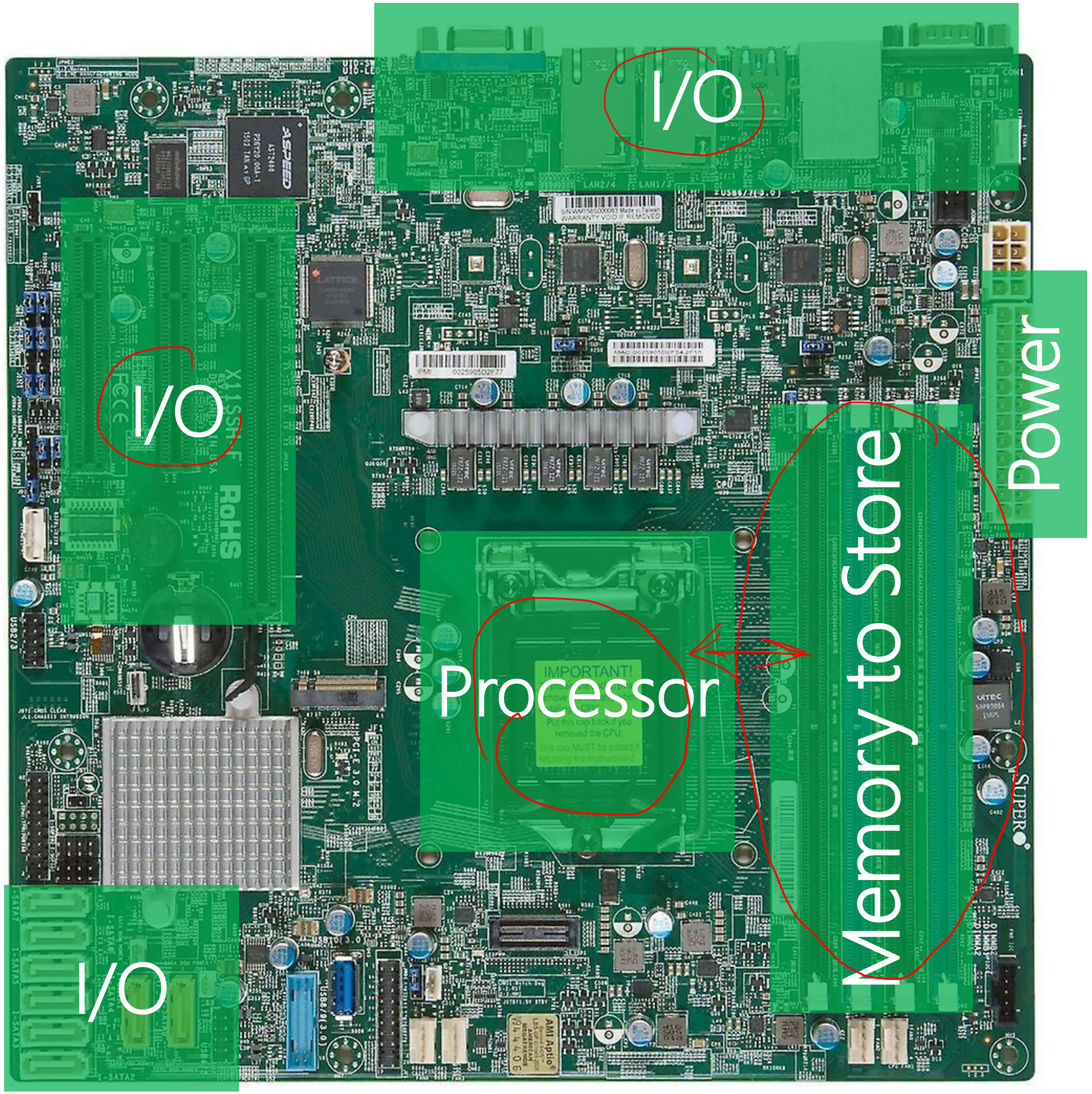
von Neumann Architecture

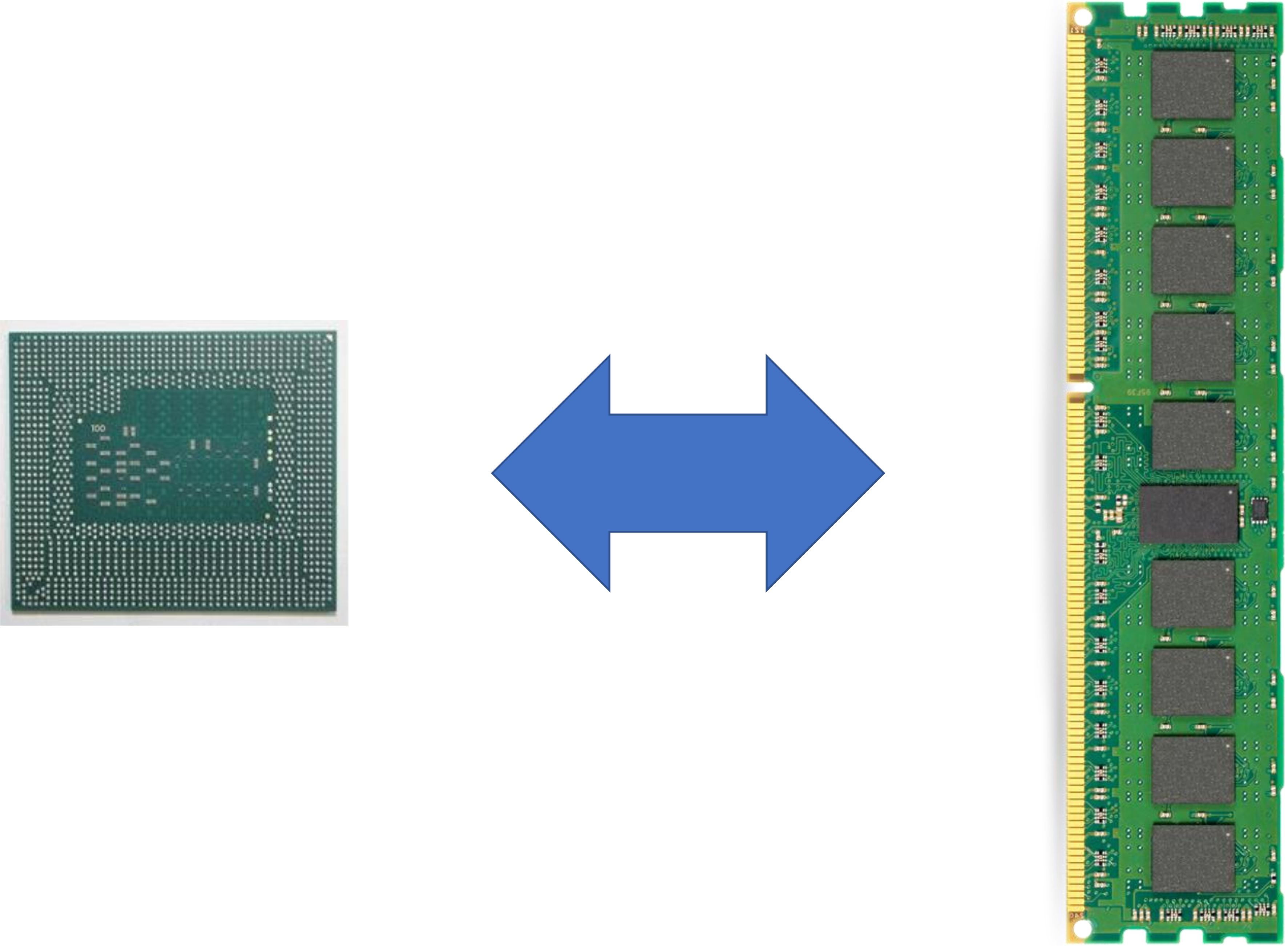
Principles

- Data and instructions are both stored in the main memory
- The content of the memory is addressable by location (regardless of what is stored in that location)
- Instructions are executed sequentially unless the order is explicitly modified


Computer System
Computer
Memory to Store
Data (Input, Output)
int a=1; int b=2; int c;
Instructions (Code)
c = a + b;
Bus
Processor
Input/Output Devices
scanf("%d", &a);
scanf("%d", &b);
printf("%d", c);
Bus
Bus
Permanent Storage
fprintf()
fscanf()
fread()
fwrite()
fseek()


IMPORTANT!
To protect the socket pins, remove this cap only when installing a CPU.
Put this cap back if you removed the CPU.
This cap MUST be placed if returning the motherboard.
X11SSH-F
RoHS
DESIGNED IN USA
LAN2/4
LAN1/3
COM1
ASPEED
SUPER


I/O
I/O
Power
Memory to Store
Processor
I/O



Data Transmission

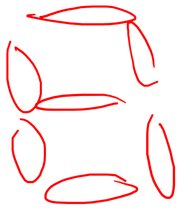
Decoder, Encoder

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

Binary Decoder

XS-3
Aiken

Binary Code Decoder



Display Decoder

7-Seg

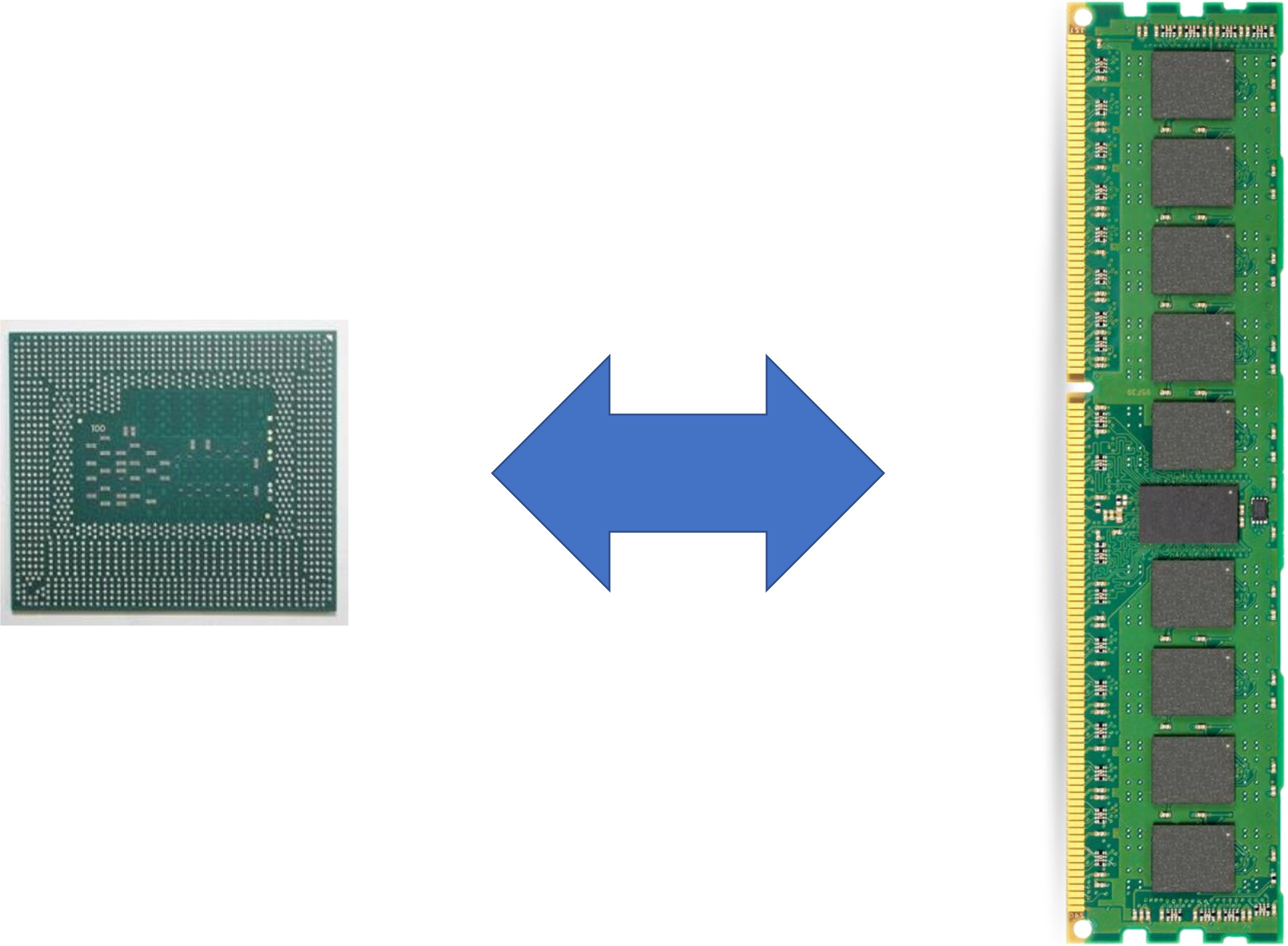
Decoder
Encode Binary to 1-hot

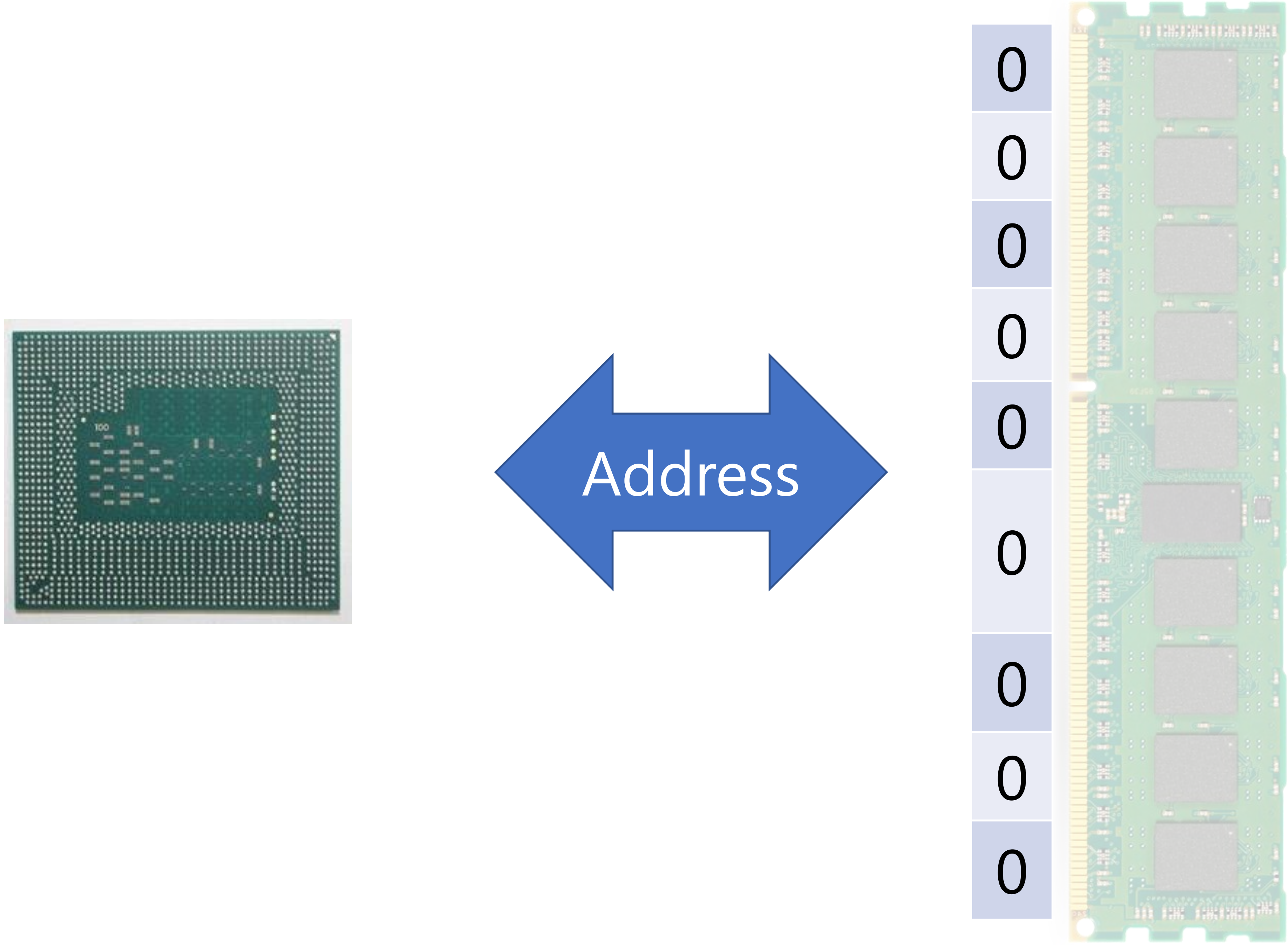
1-hot: a vector of bits with a single 1 and all the others 0

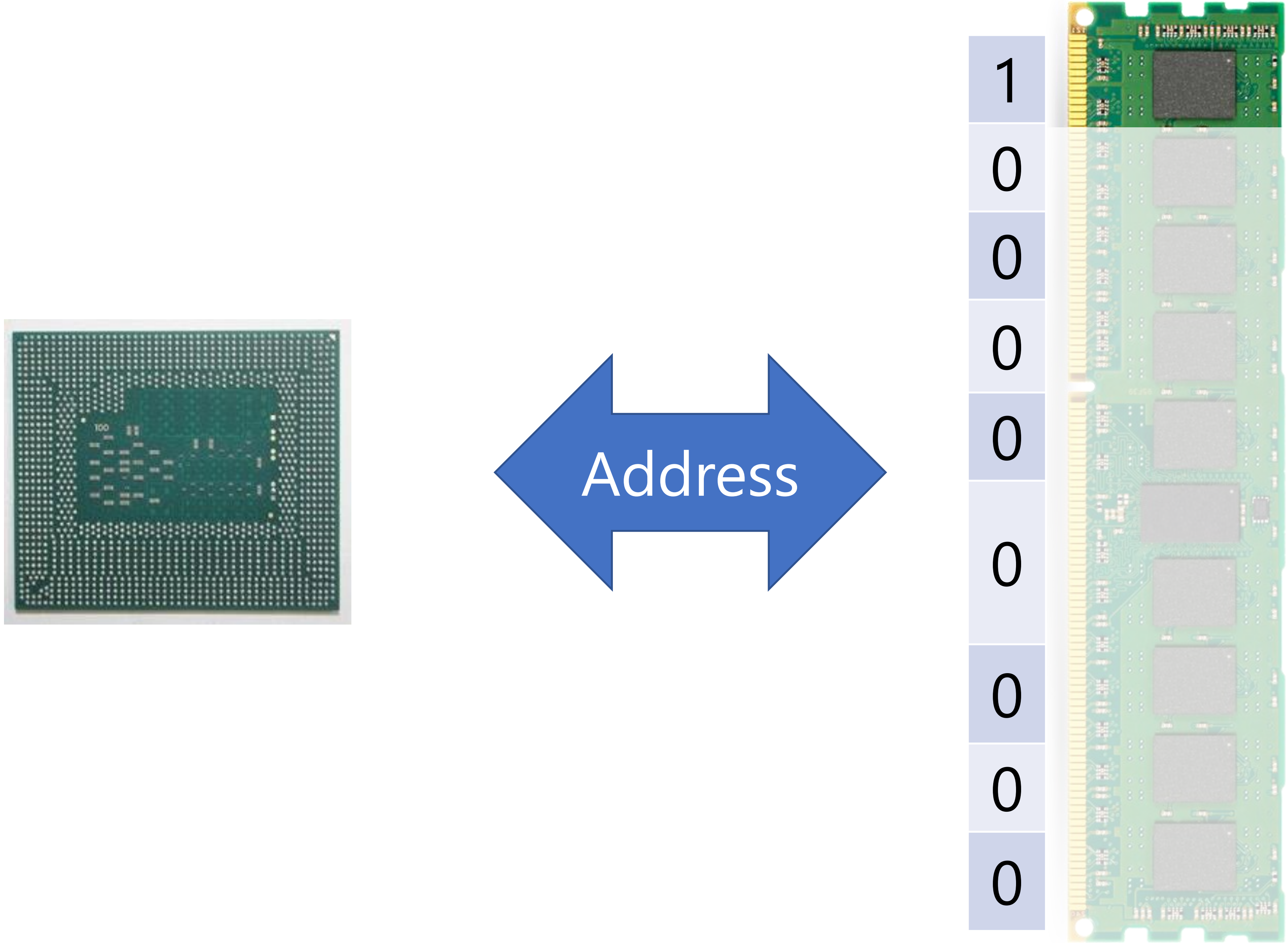
[0010000000]

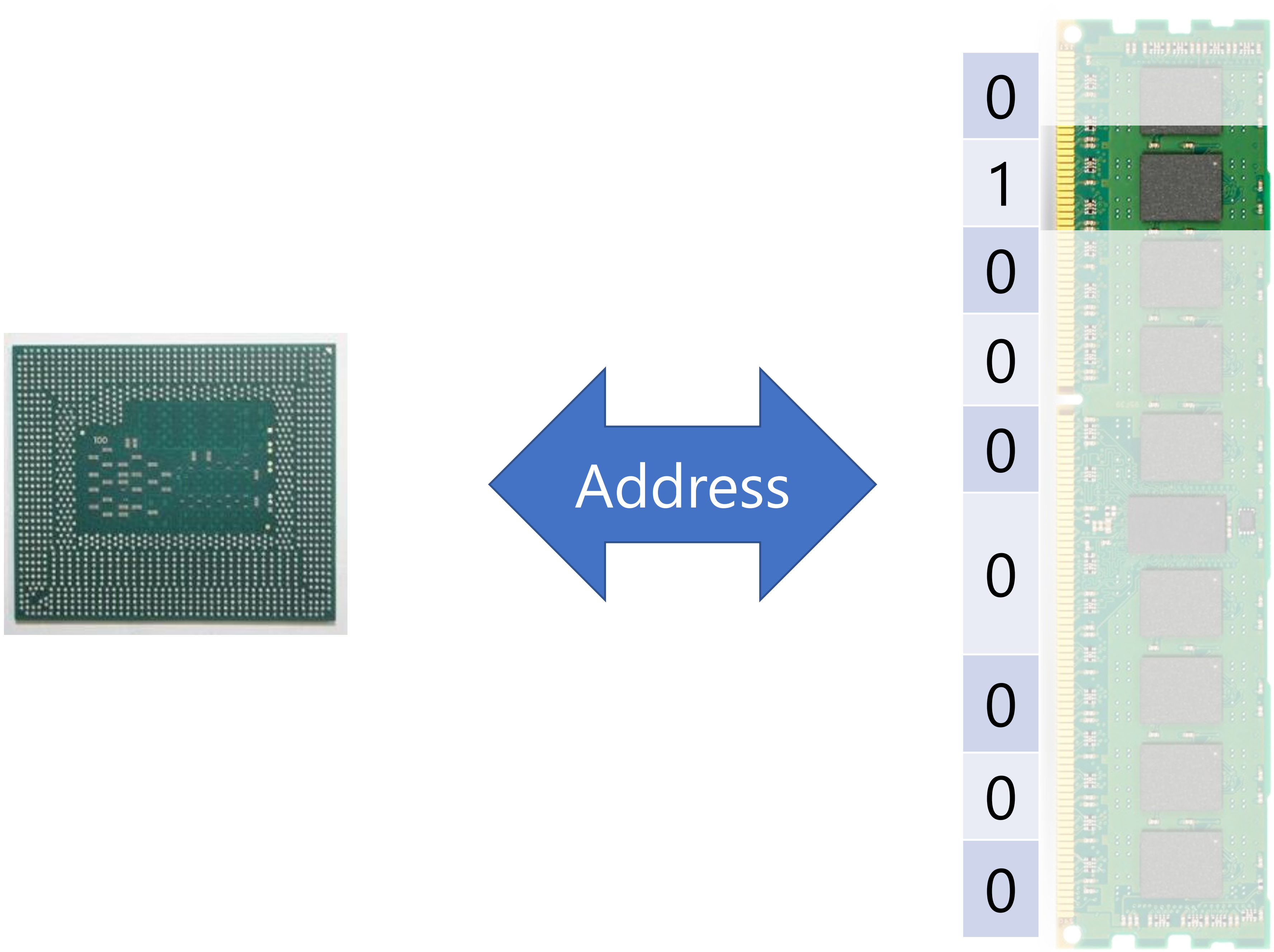
[0000000100]

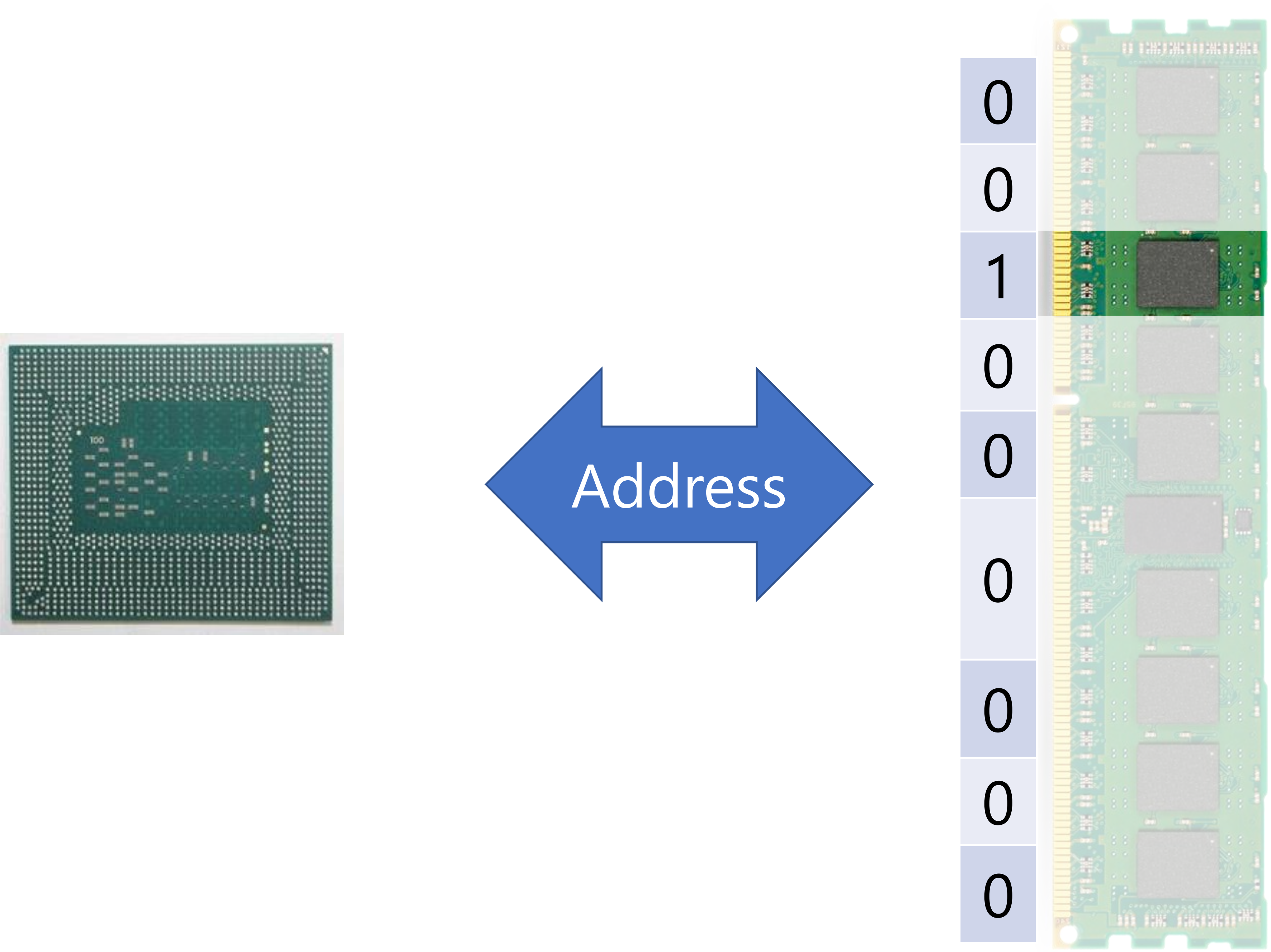
~~[0010010000]~~

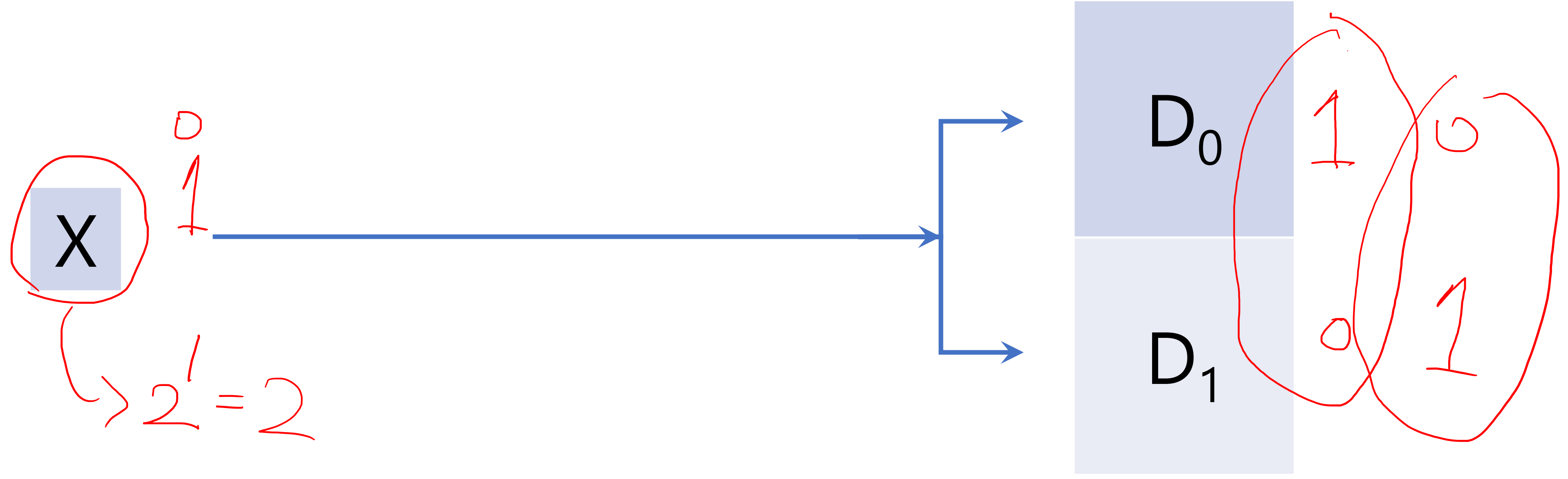



Address
0
0
0
0
0
0
0
0
0


Address
1
0
0
0
0
0
0
0
0
0


Address
0
1
0
0
0
0
0
0
0

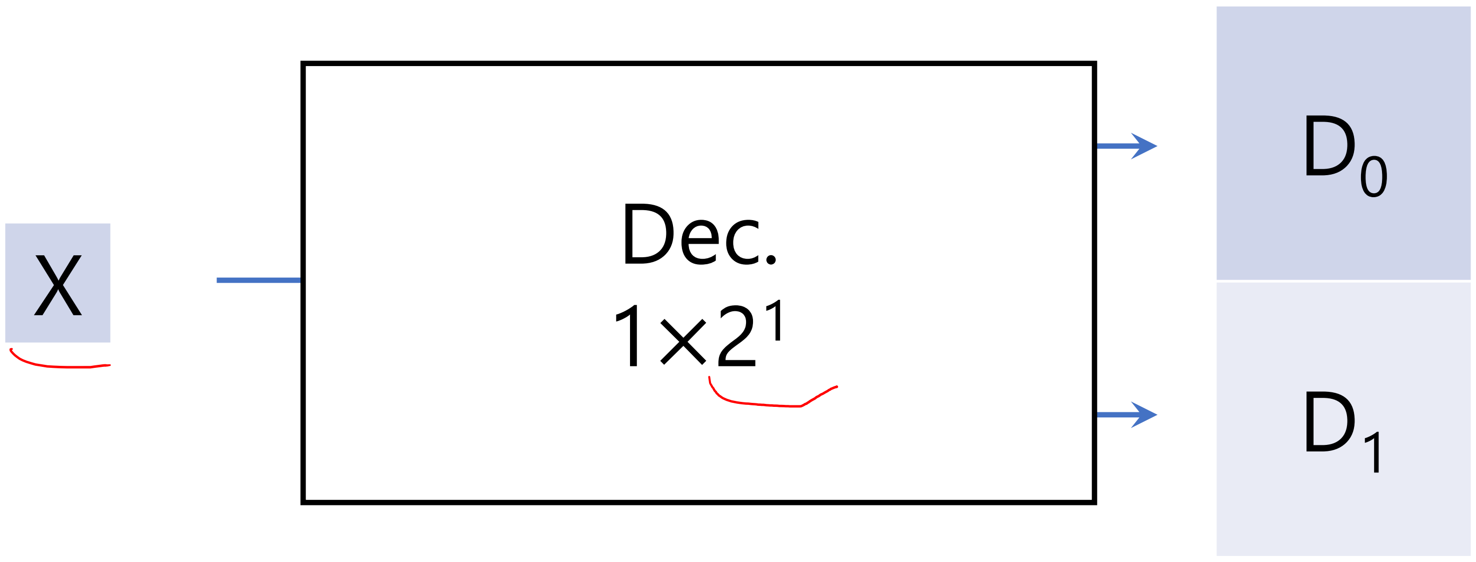

Address
0
0
1
0
0
0
0
0
0



X
0
1
$2^1 = 2$
D0
D1
1
0
0
1


0
$D_0=1$
$D_1=0$

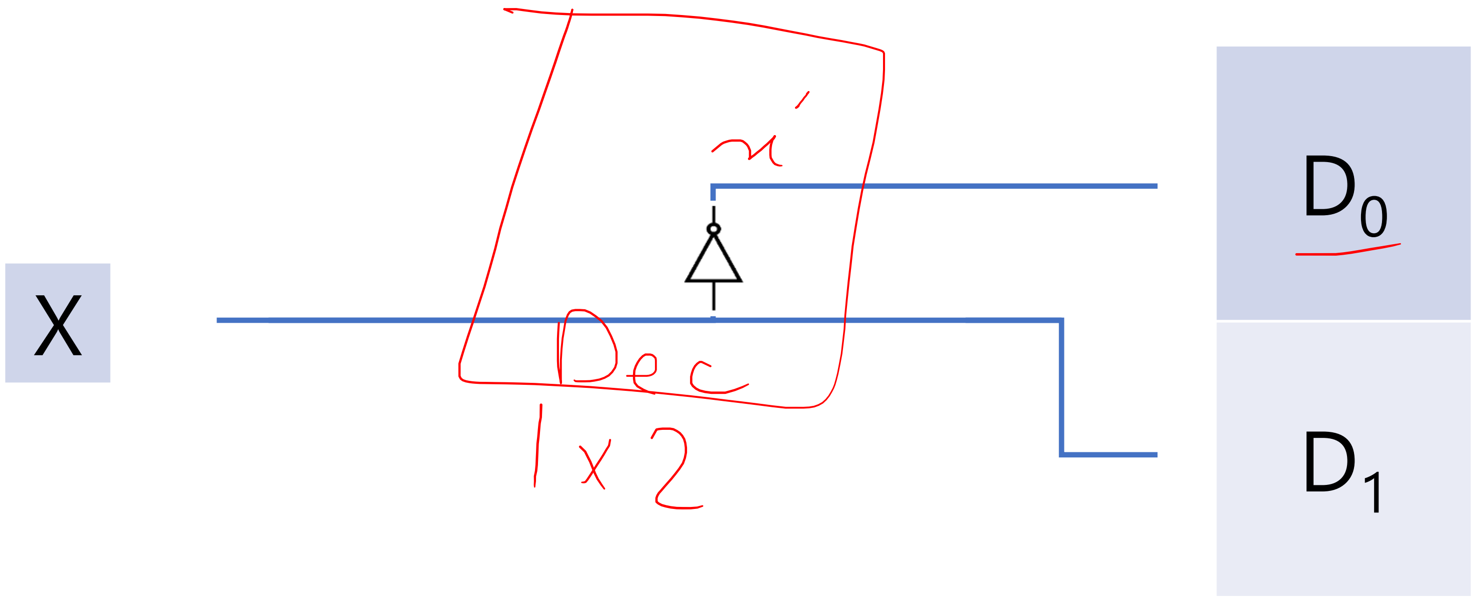

1
$D_0=0$
$D_1=1$

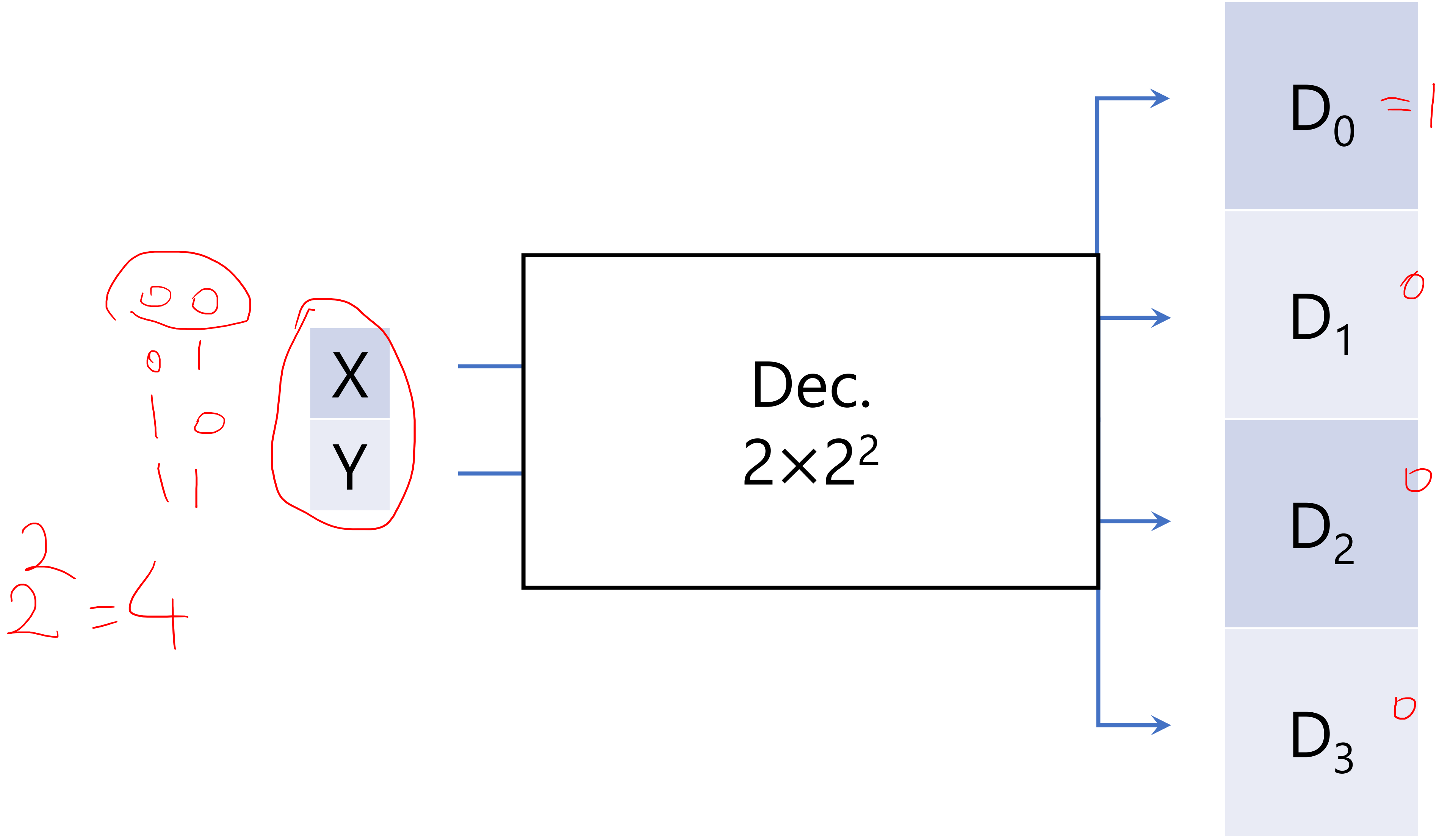

X
Dec.
1×2^1
D0
D1

F_1 F_2

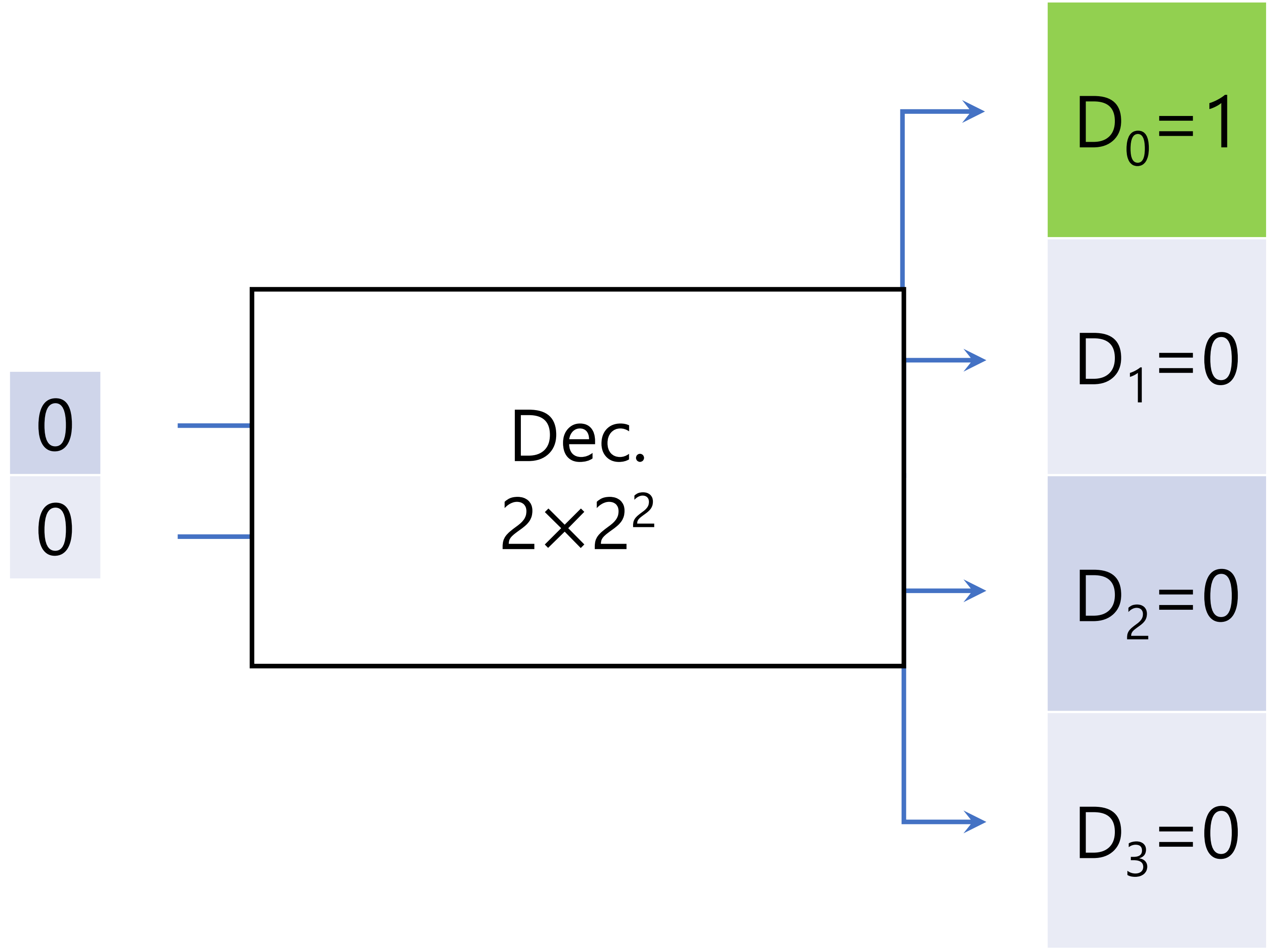
X	$D_0 = m_0$	$D_1 = m_1 = M_0$
0	1	0
1	0	1

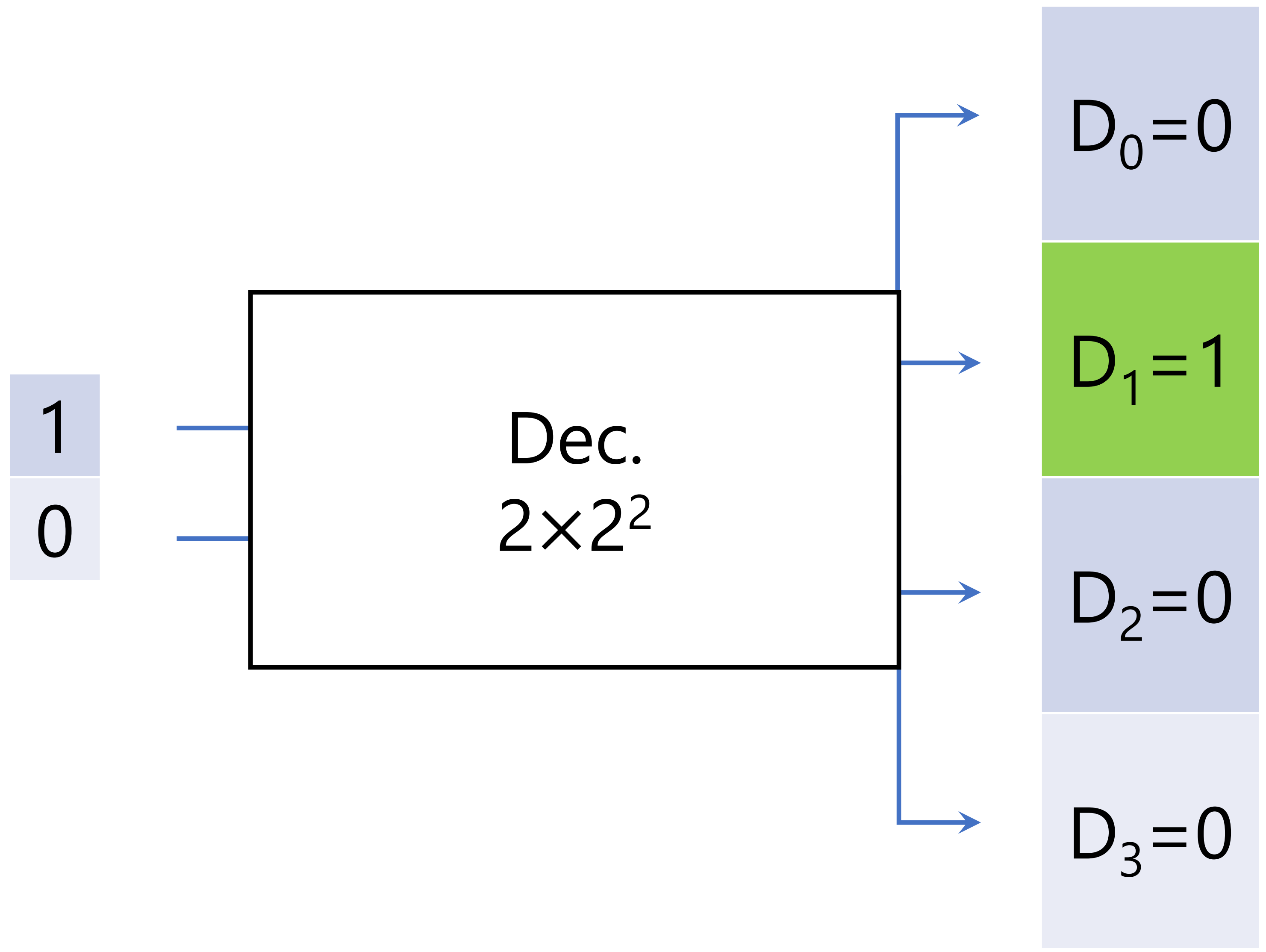
X	$D_0 = m_0 = X'$	$D_1 = m_1 = X$
0	1	0
1	0	1

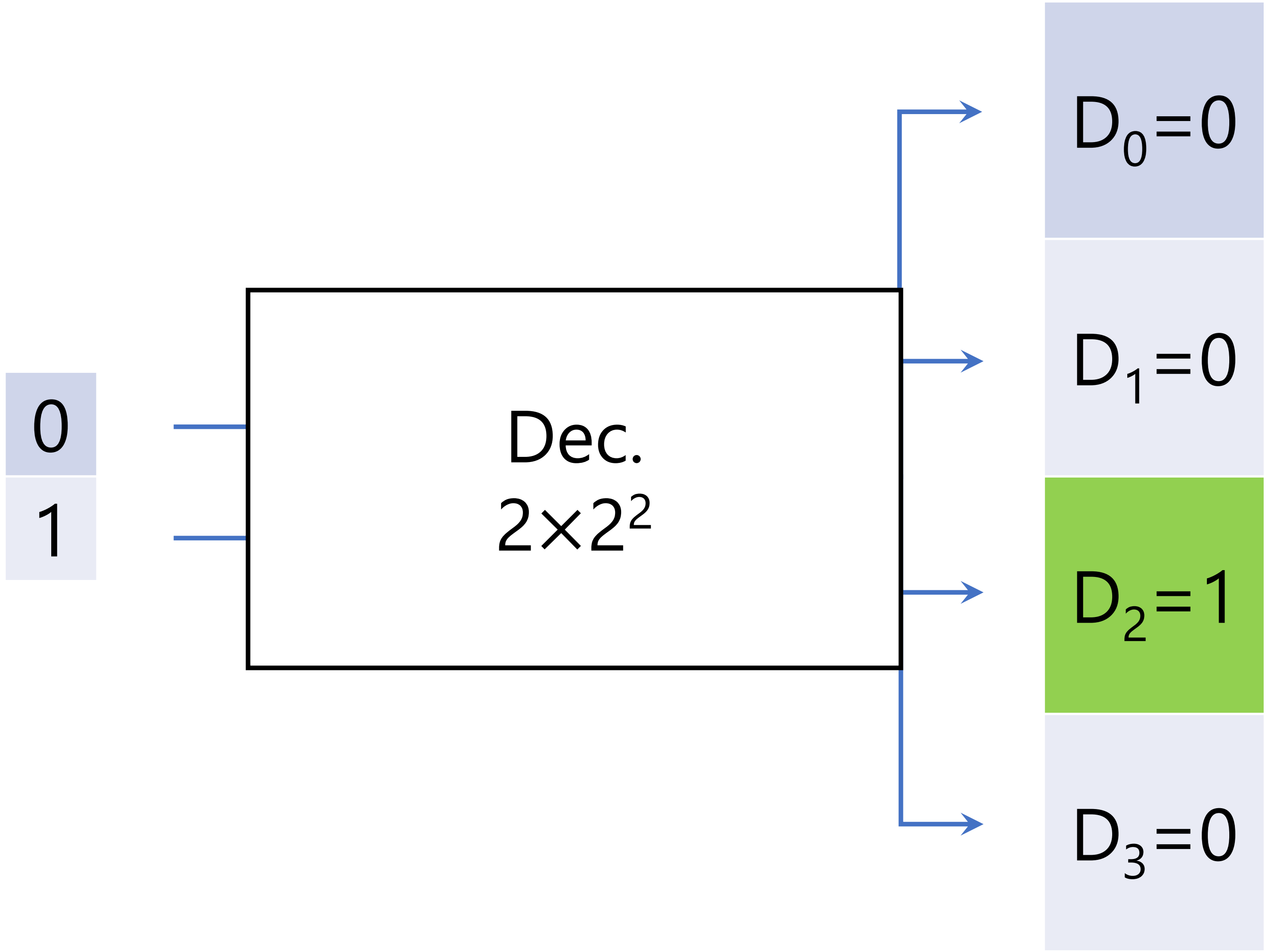

X
x'
Dec
1x2
D0
D1



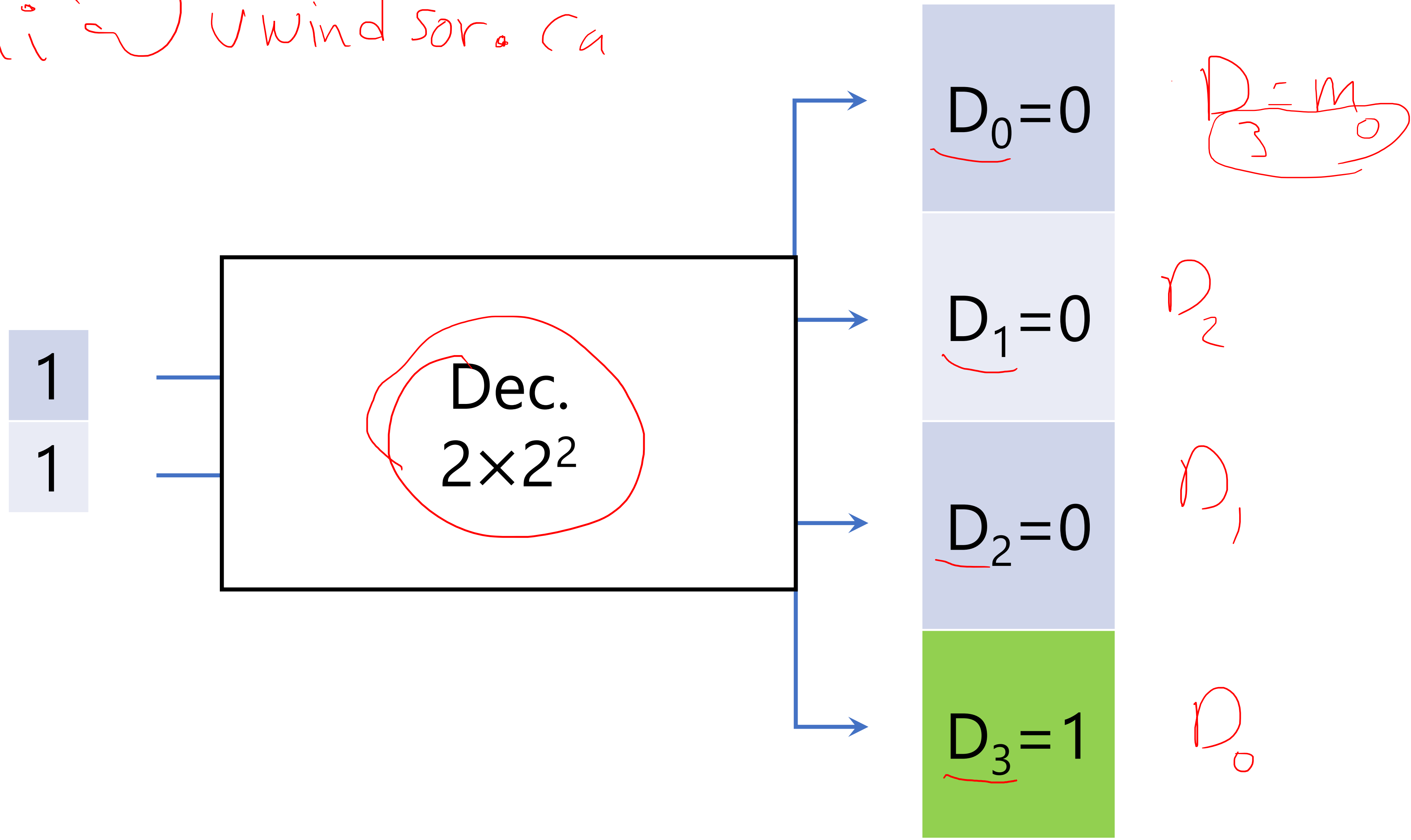
X
Y
Dec.
2×2²
D0 =1
D1 0
D2 0
D3 0
0 0
0 1
1 0
1 1
2² = 4


0
0
Dec.
2×2²
D₀=1
D₁=0
D₂=0
D₃=0


1
0
Dec.
2×2²
D₀=0
D₁=1
D₂=0
D₃=0


0
1
Dec.
2×2²
D₀=0
D₁=0
D₂=1
D₃=0

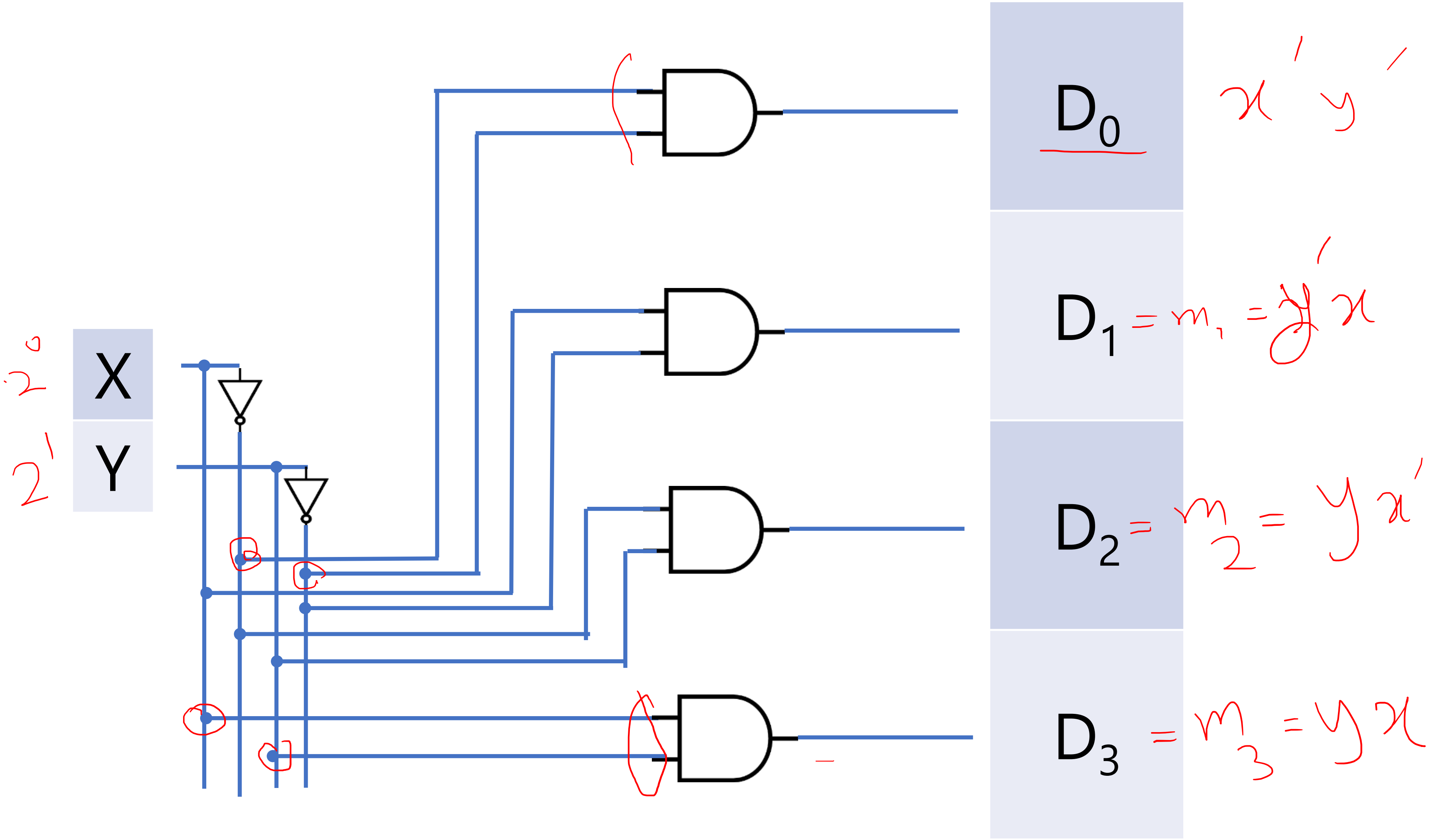
hfani@uwindsor.ca



$D_i = m_i$

sop

Y	X	$D_0 = m_0$	$D_1 = m_1$	$D_2 = m_2$	$D_3 = m_3$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



2^0
X
2^1
Y
D_0
$x' y'$
$D_1 = m_1 = y' x$
$D_2 = m_2 = y x'$
$D_3 = m_3 = y x$

3×2^3

$D_i = m_i$

Table 4.6
Truth Table of a Three-to-Eight-Line Decoder

$= m_0 = x'y'z'$

Inputs			Outputs							
x	y	z	D_0	D_1 $= m_1$	D_2 m_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$x'y'z$

$x'yz'$

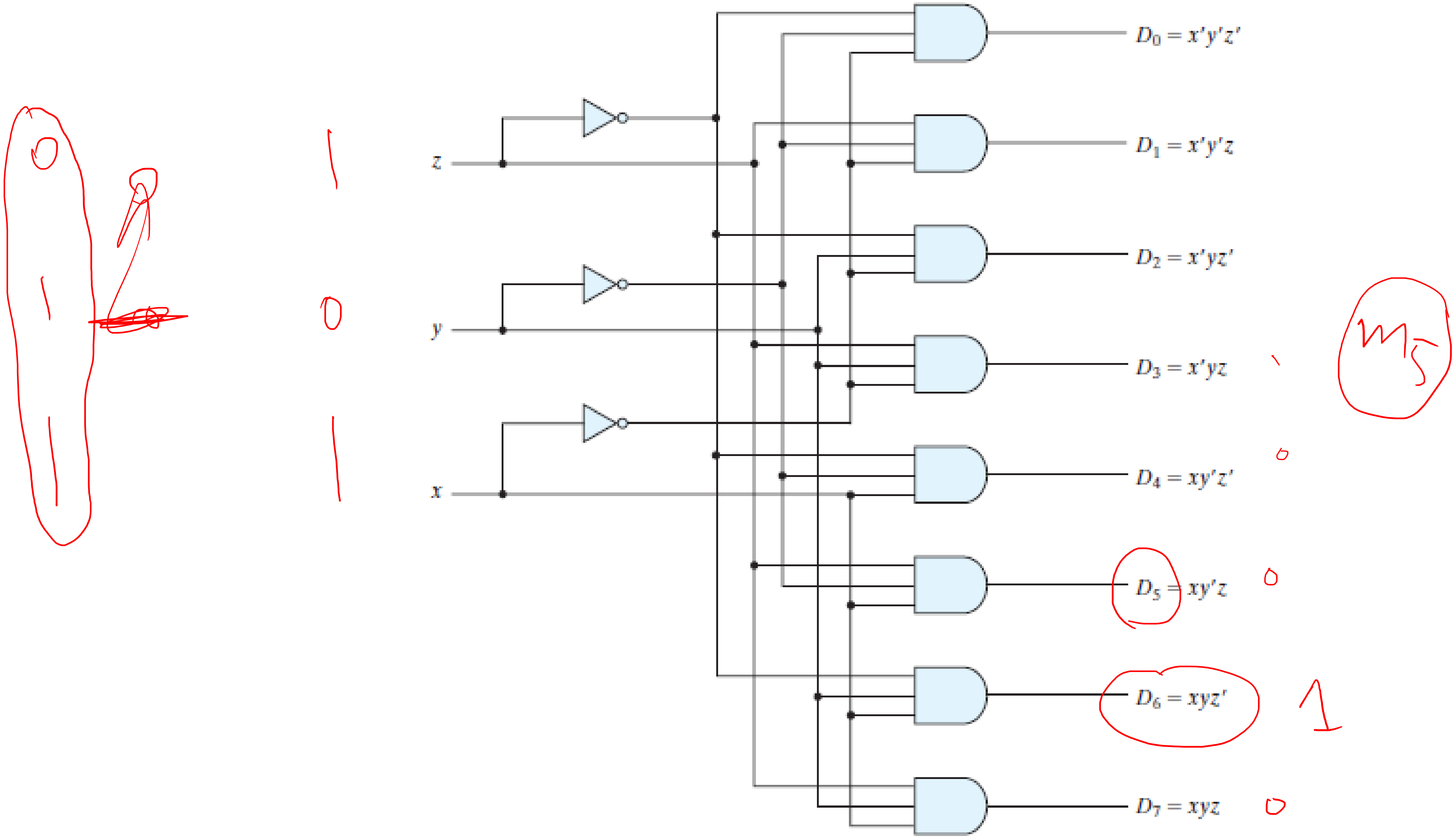


FIGURE 4.18
Three-to-eight-line decoder

Decoder

Encode 4-Bit Binary to 2^4 One-hot

Decoder

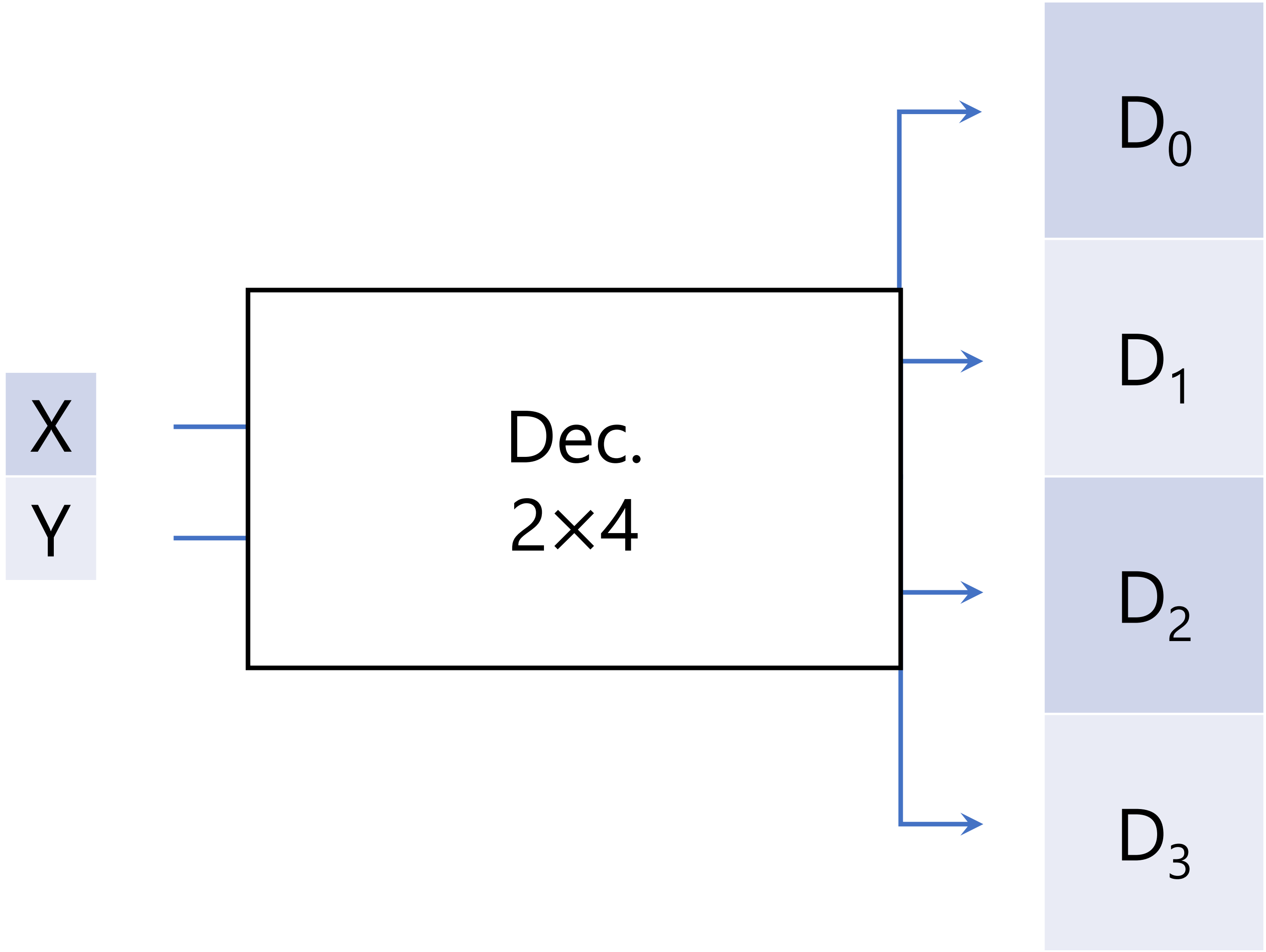
Encode n-Bit Binary to 2^n One-hot

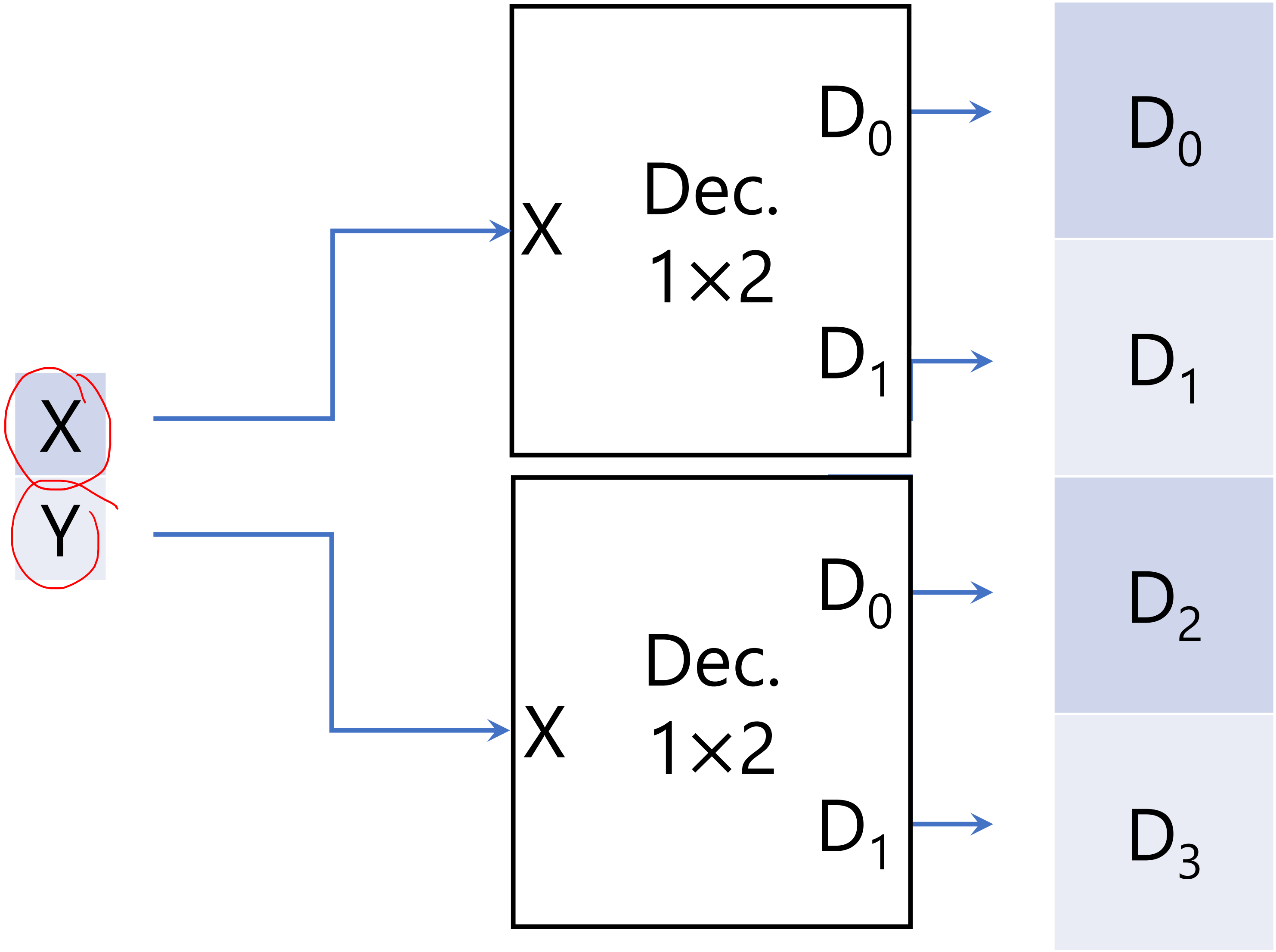
2×2^4 $\leftarrow$ 4×2^4 $\leftarrow$ 5×2^5

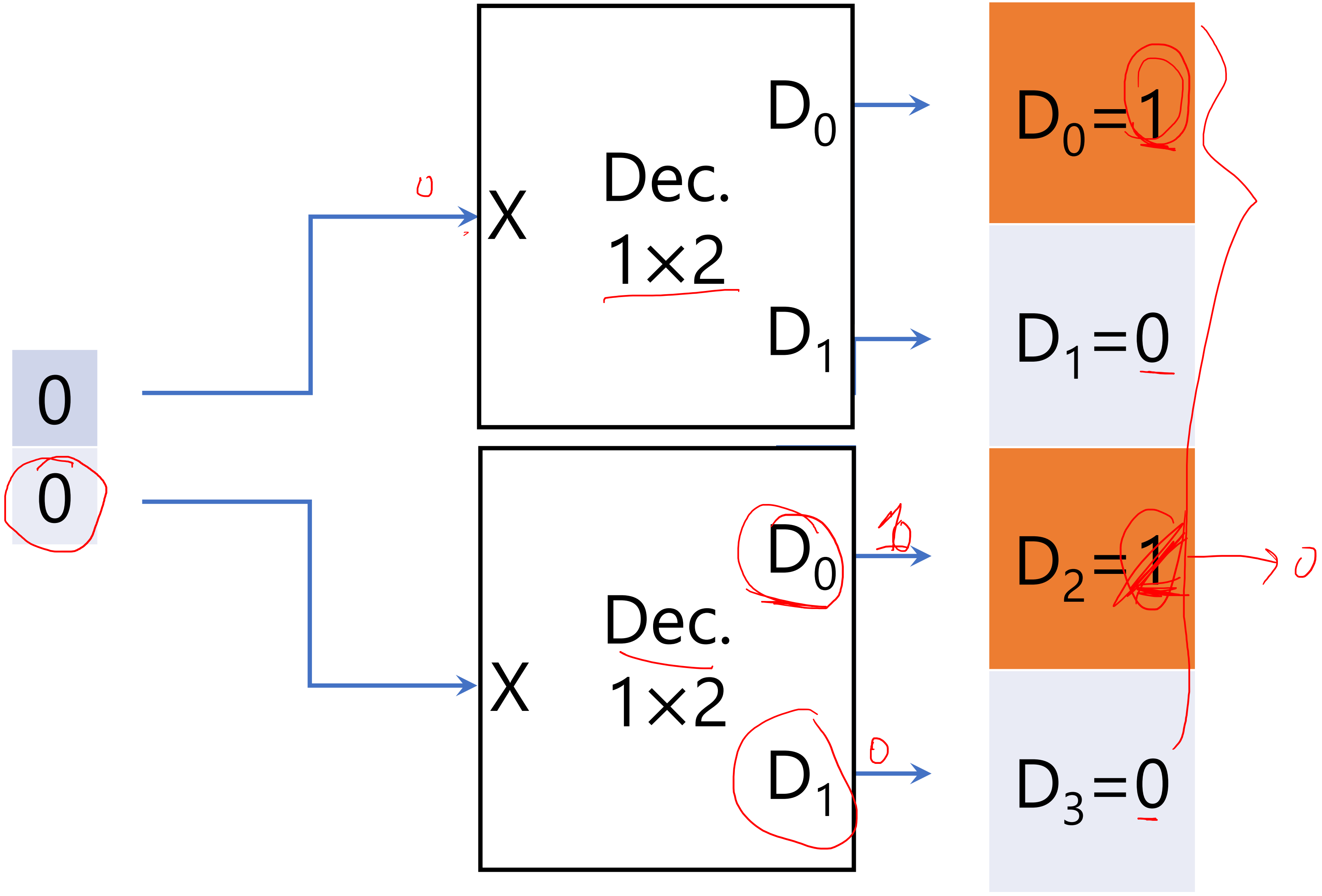
Decoder

Encode 2-Bit Binary to 2^2 One-hot

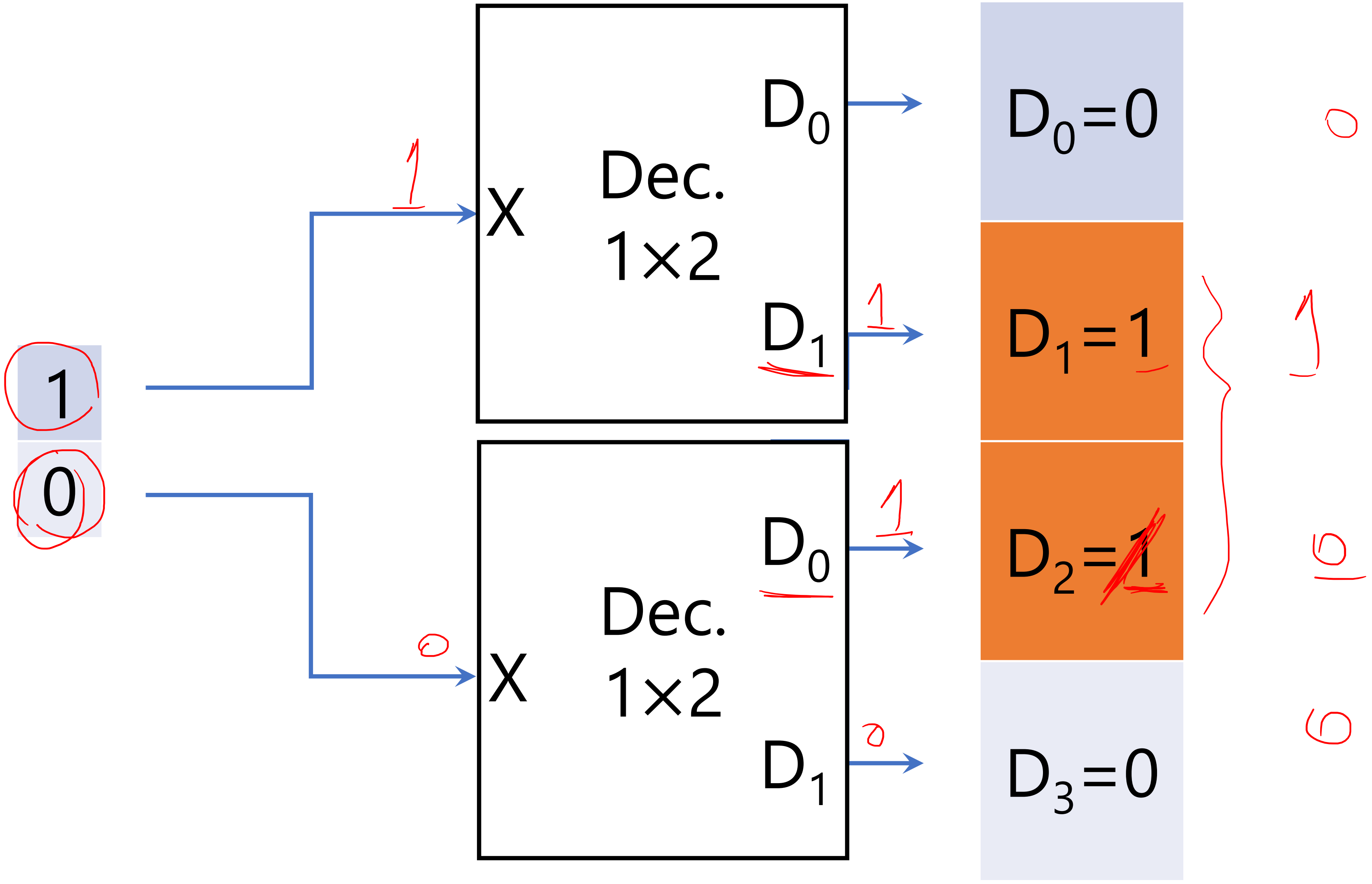
Re-Use 1×2^1 Decoder


X
Y
Dec.
2×4
D0
D1
D2
D3


X
Y
X
Dec.
1×2
D0
D1
X
Dec.
1×2
D0
D1
D0
D1
D2
D3

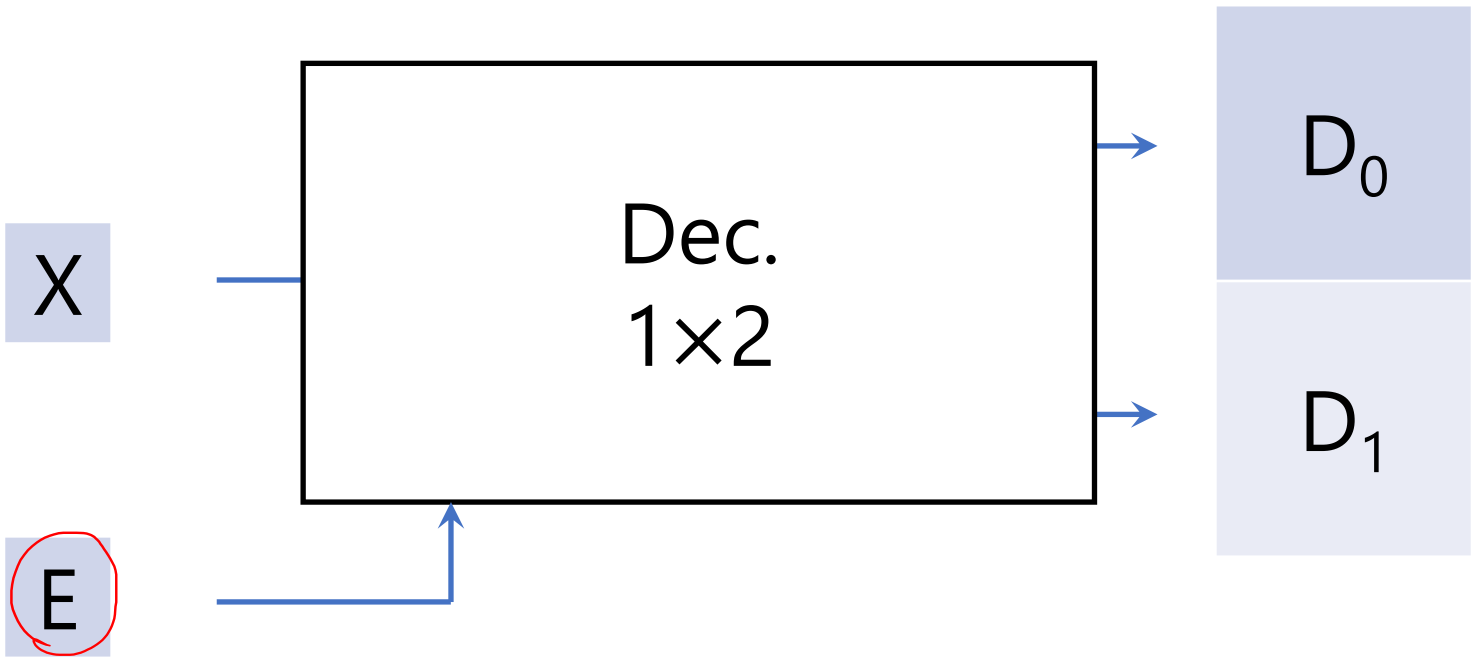


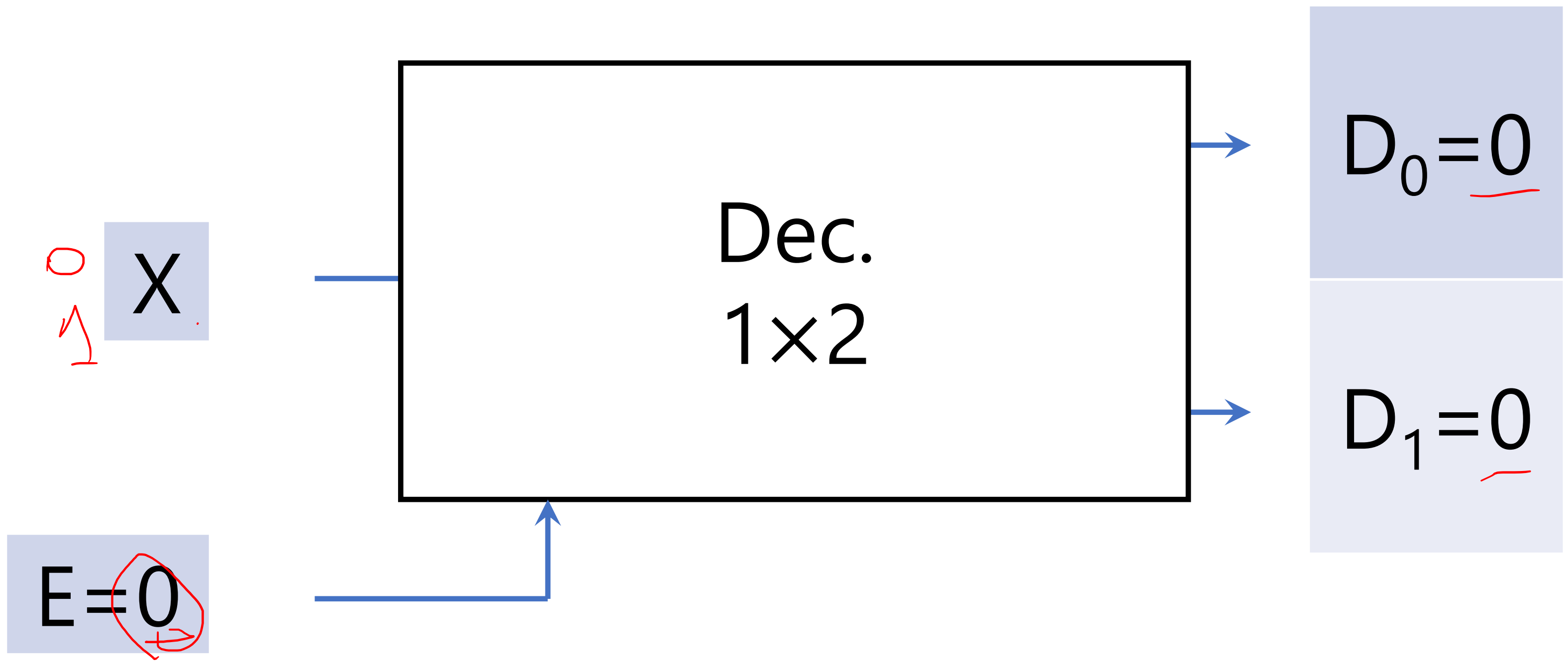
0
0
X
Dec.
1×2
D0
D1
X
Dec.
1×2
D0
D1
D0=1
D1=0
D2=1
D3=0



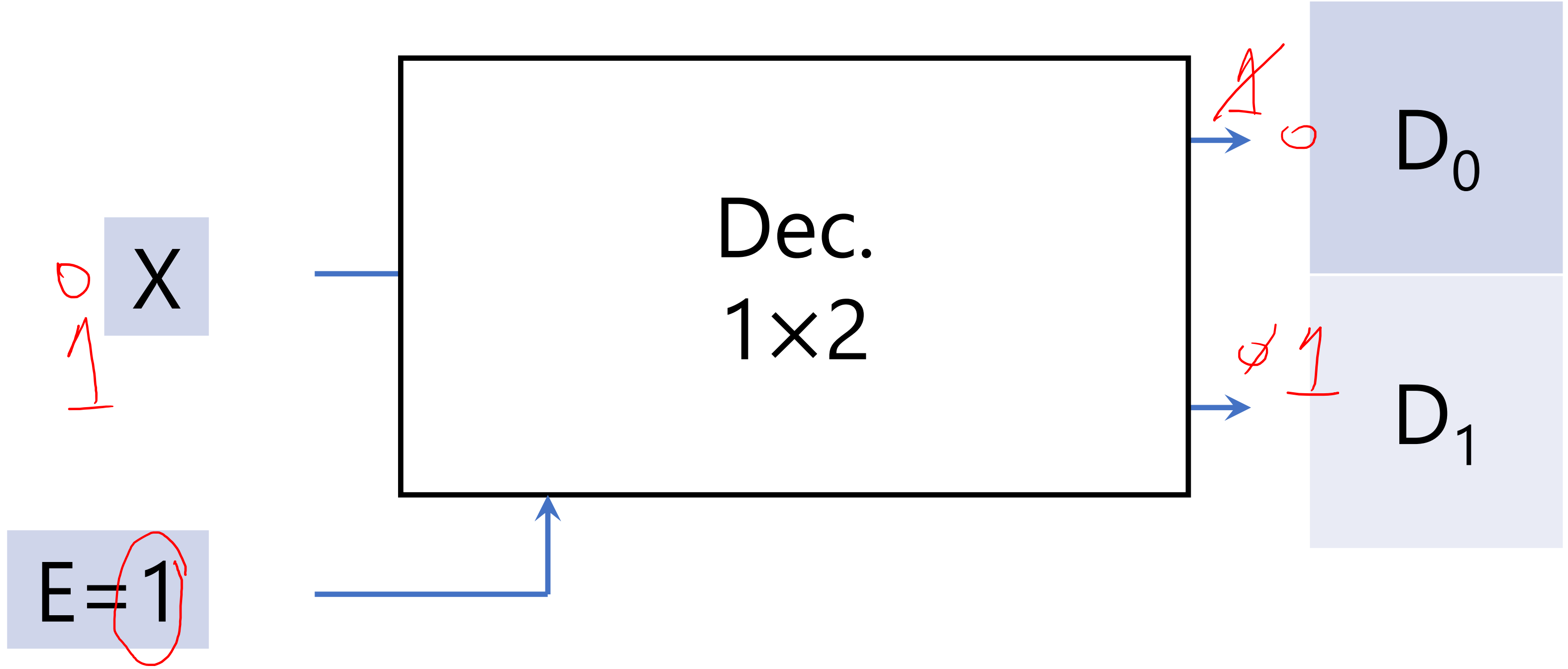
1
0
1
X
Dec.
1×2
D0
D1
1
0
X
Dec.
1×2
D0
D1
1
0
D0=0
D1=1
D2=~~1~~
D3=0
0
1
0
0

Decoder
Enable input


X
E
Dec.
1×2
D0
D1



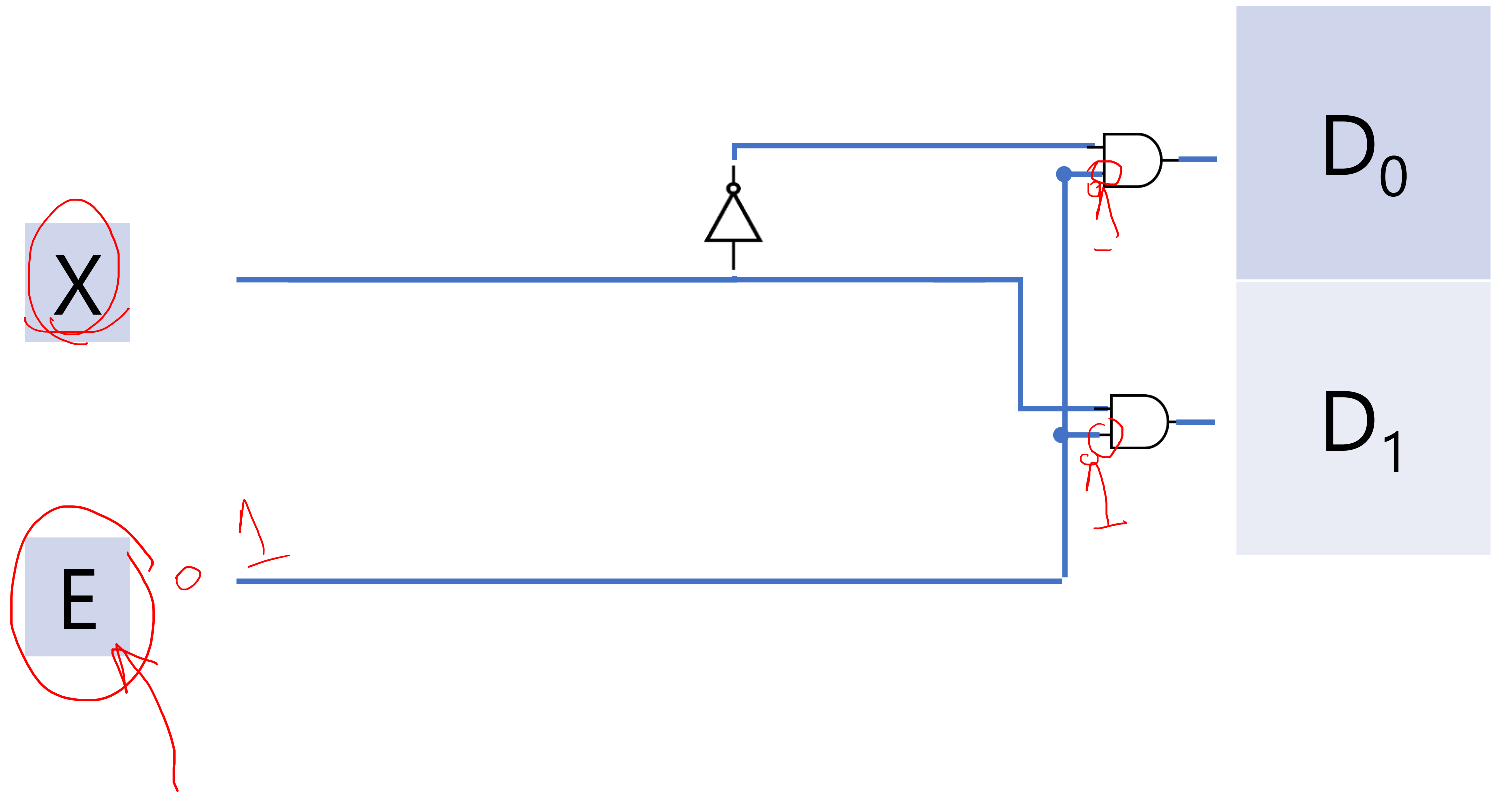
0
1
X
E=0
Dec.
1×2
D0=0
D1=0



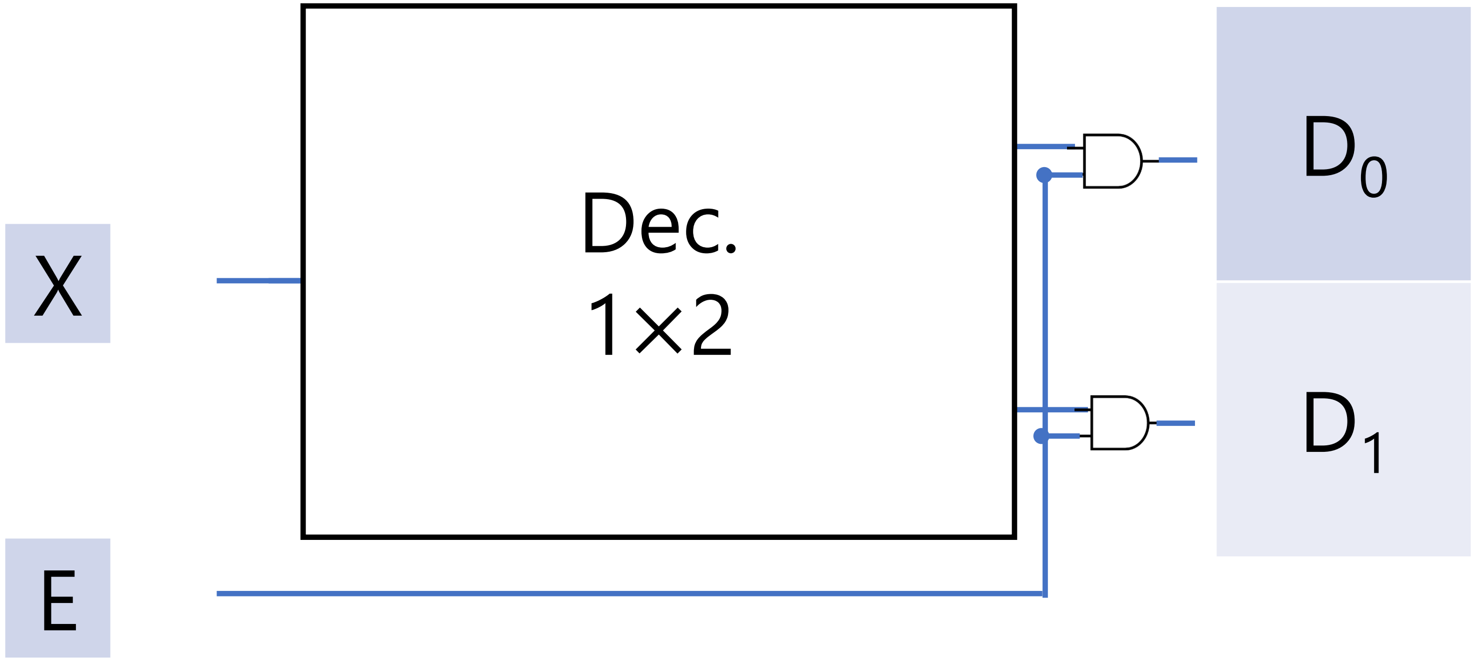
X
E=1
Dec.
1×2
D_0
D_1
0
1
0
1

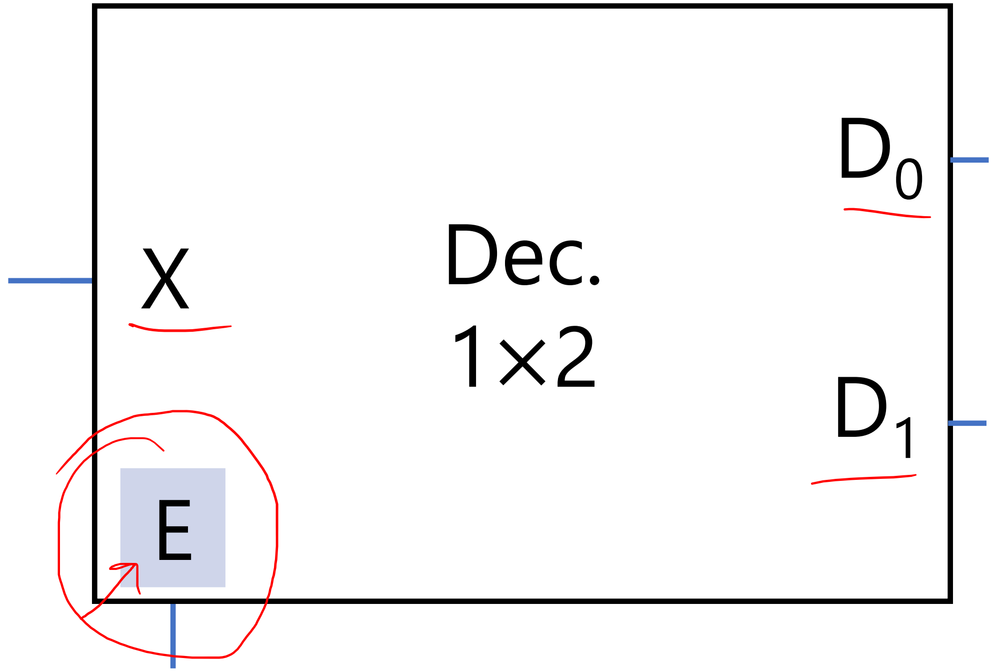
E	X	$D_0 = m_2$	$D_1 = m_3$
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

E	X	$D_0=m_0$	$D_1=m_1$
1	0	1	0
1	1	0	1

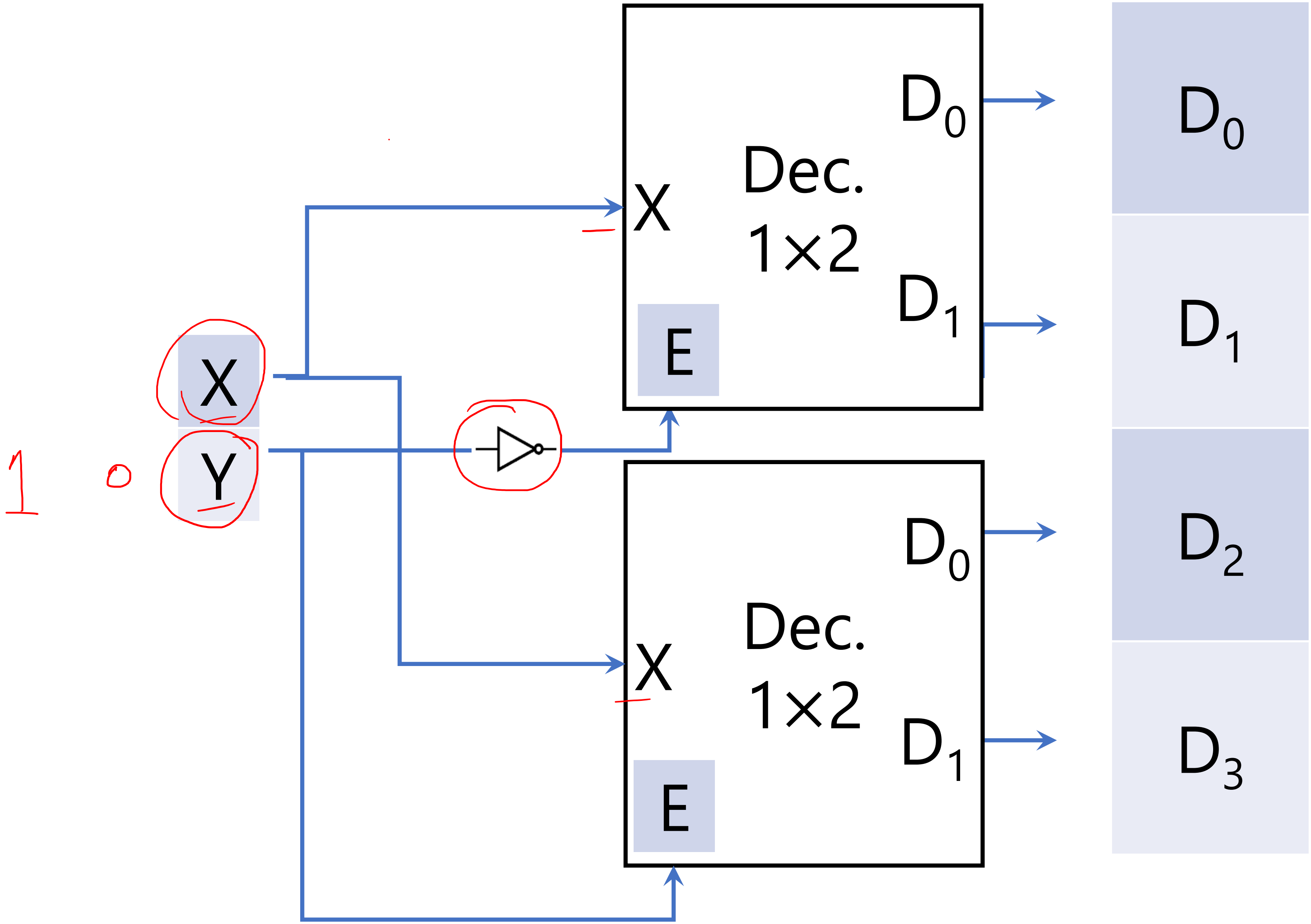


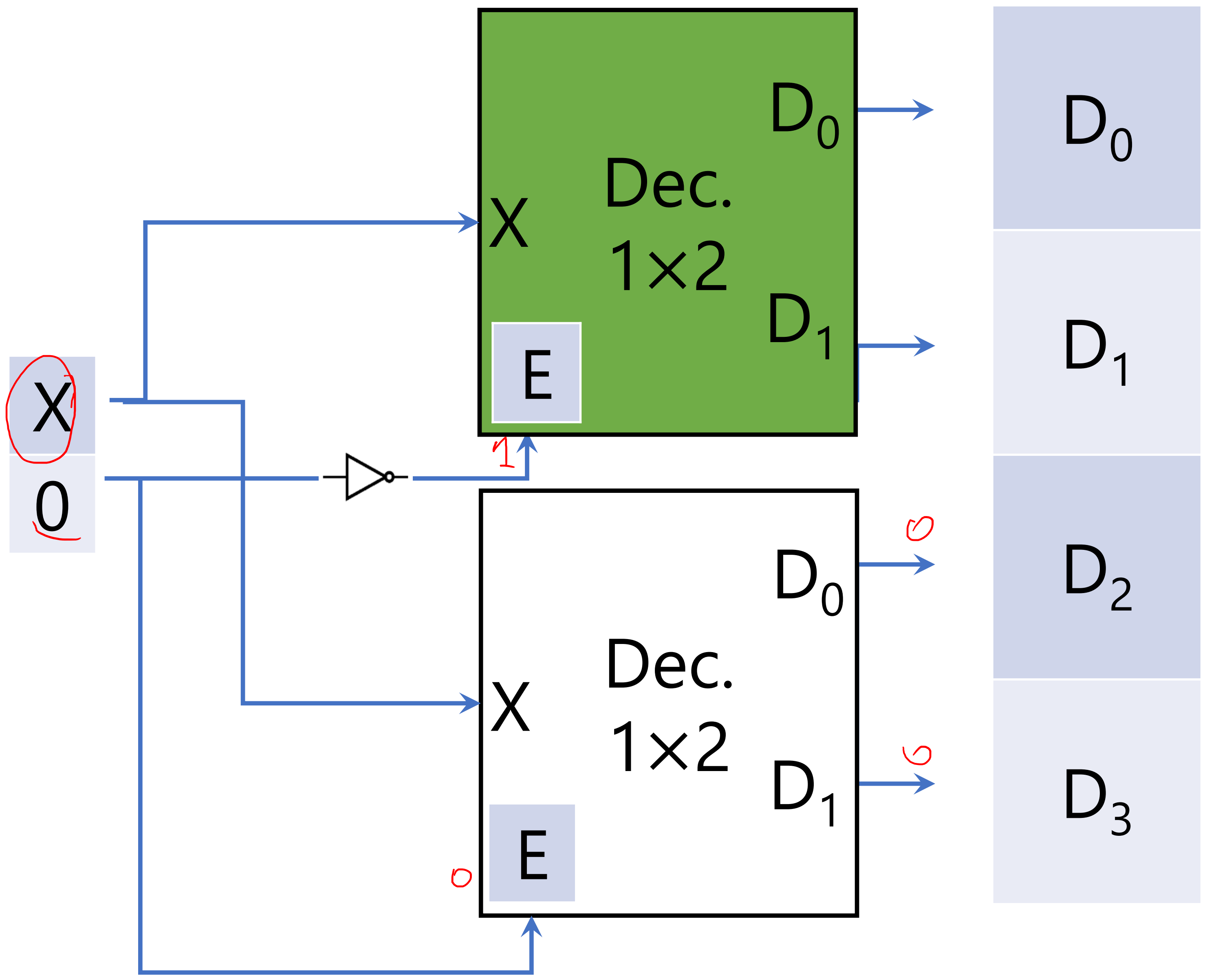
X
E
D_0
D_1


X
E
Dec.
1×2
D_0
D_1

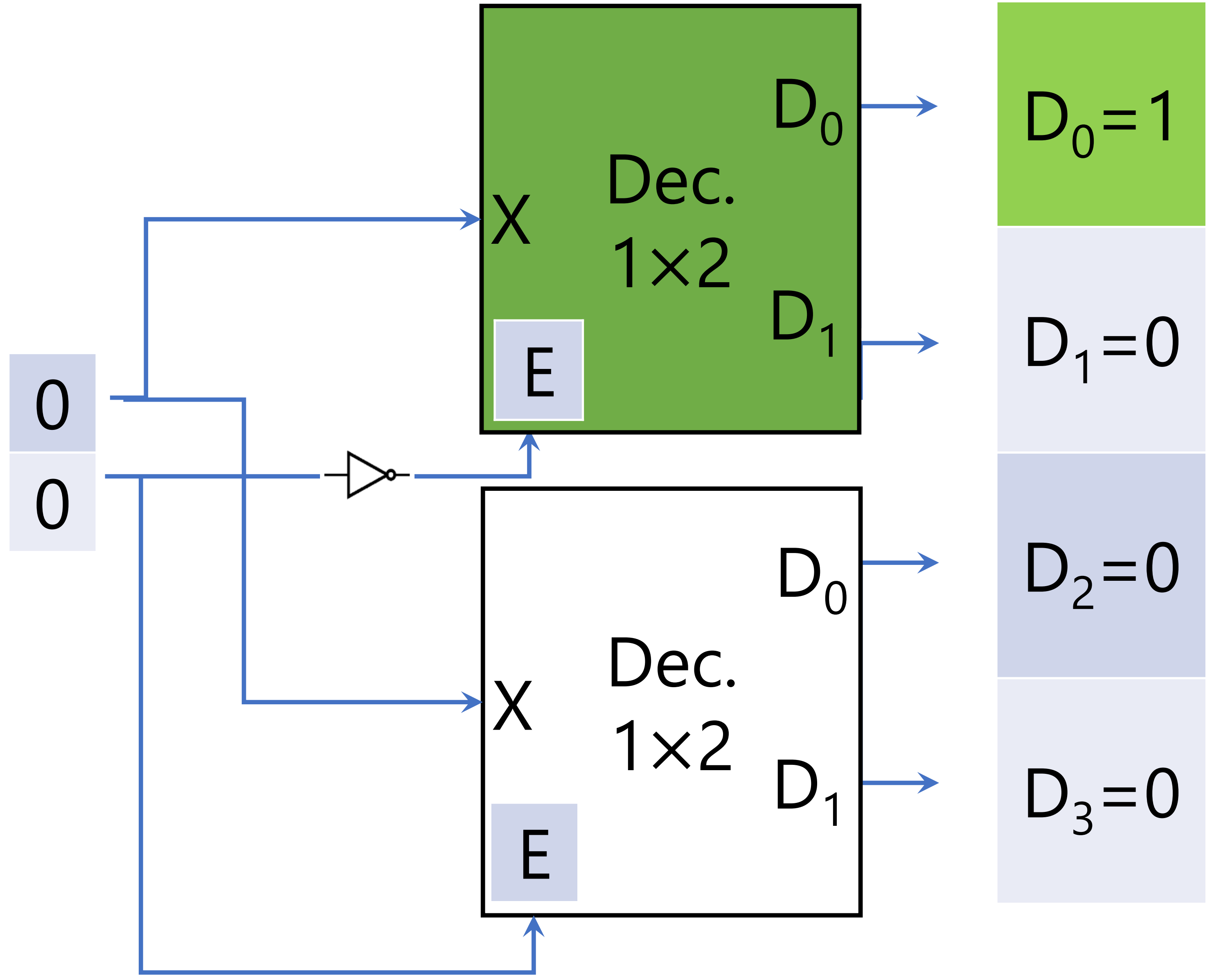


X
Dec.
1×2
E
D0
D1

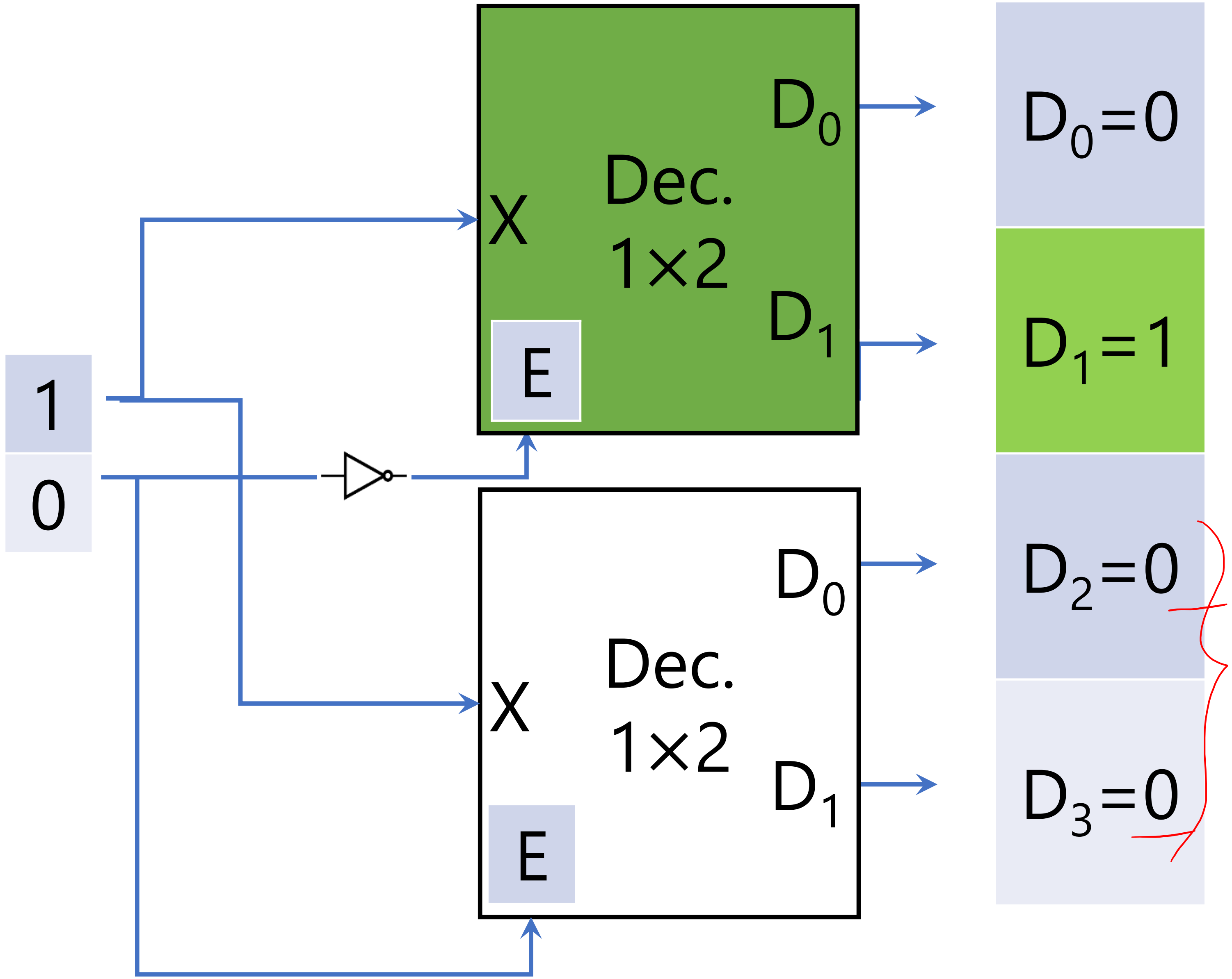

X
Y
1
0
Dec.
1×2
X
E
D0
D1
Dec.
1×2
X
E
D0
D1
D0
D1
D2
D3



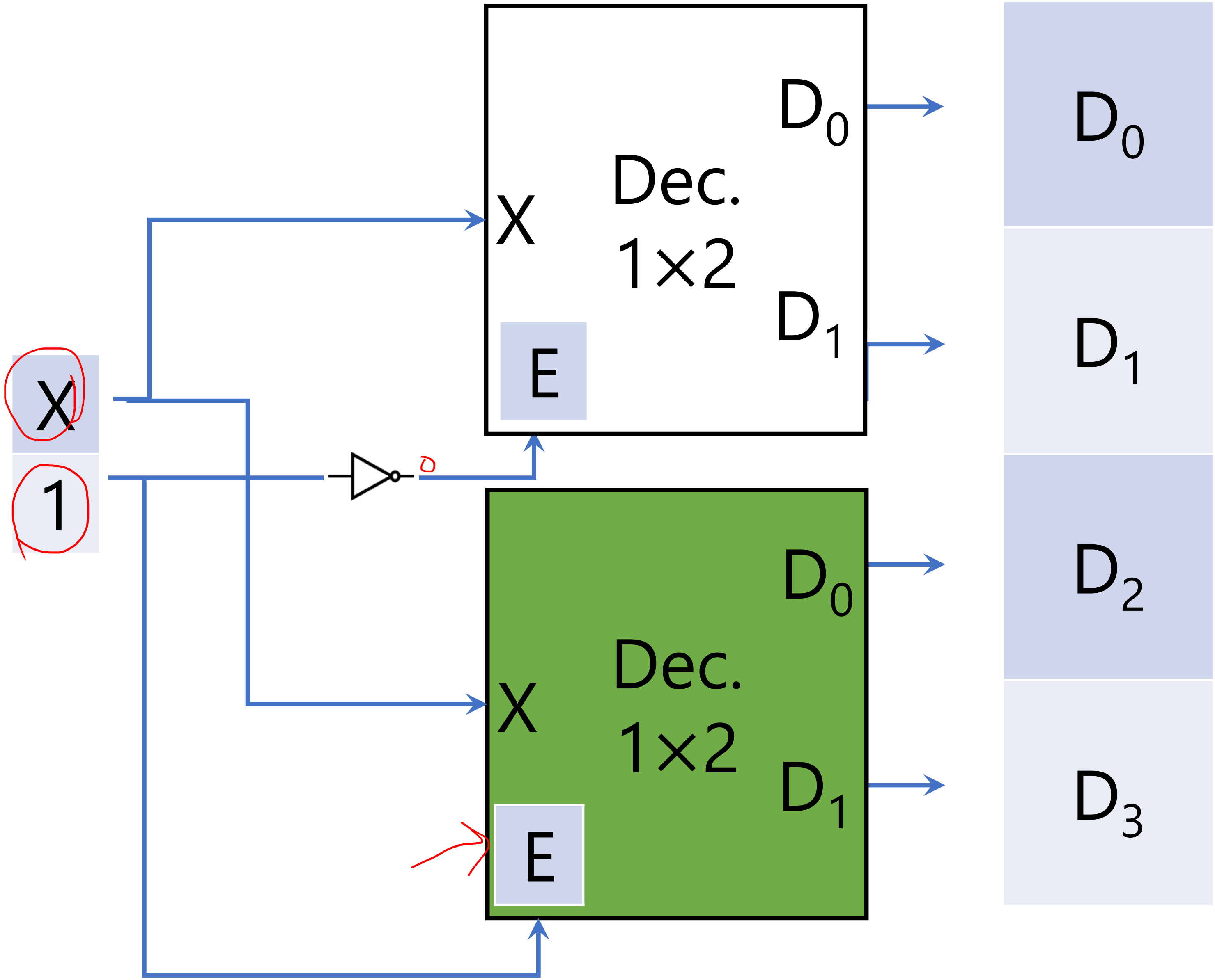
X
0
X
Dec.
1×2
D0
D1
E
1
X
Dec.
1×2
D0
D1
E
0
0
0
D0
D1
D2
D3

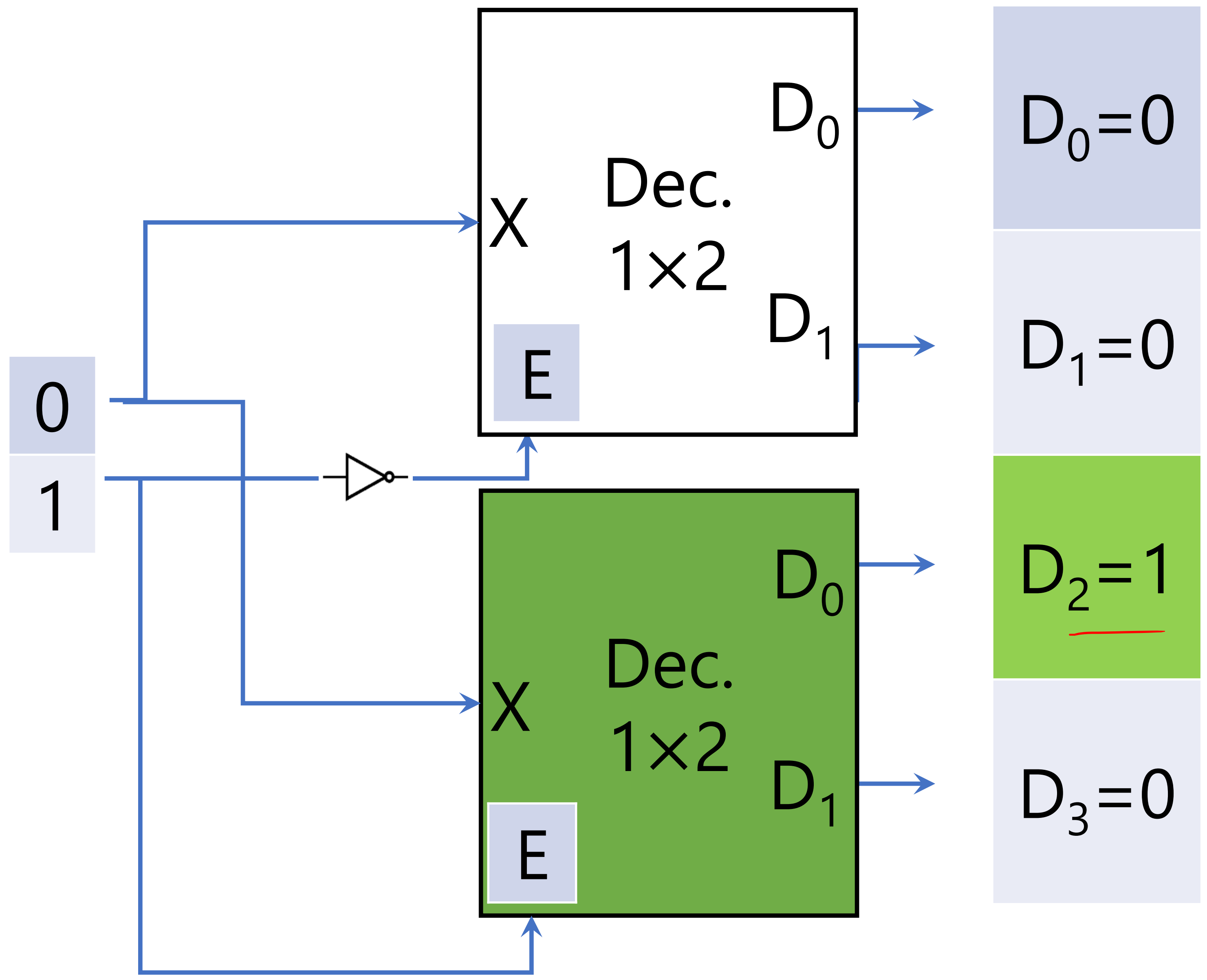


0
0
X
Dec.
1×2
E
D0
D1
X
Dec.
1×2
E
D0
D1
D0=1
D1=0
D2=0
D3=0

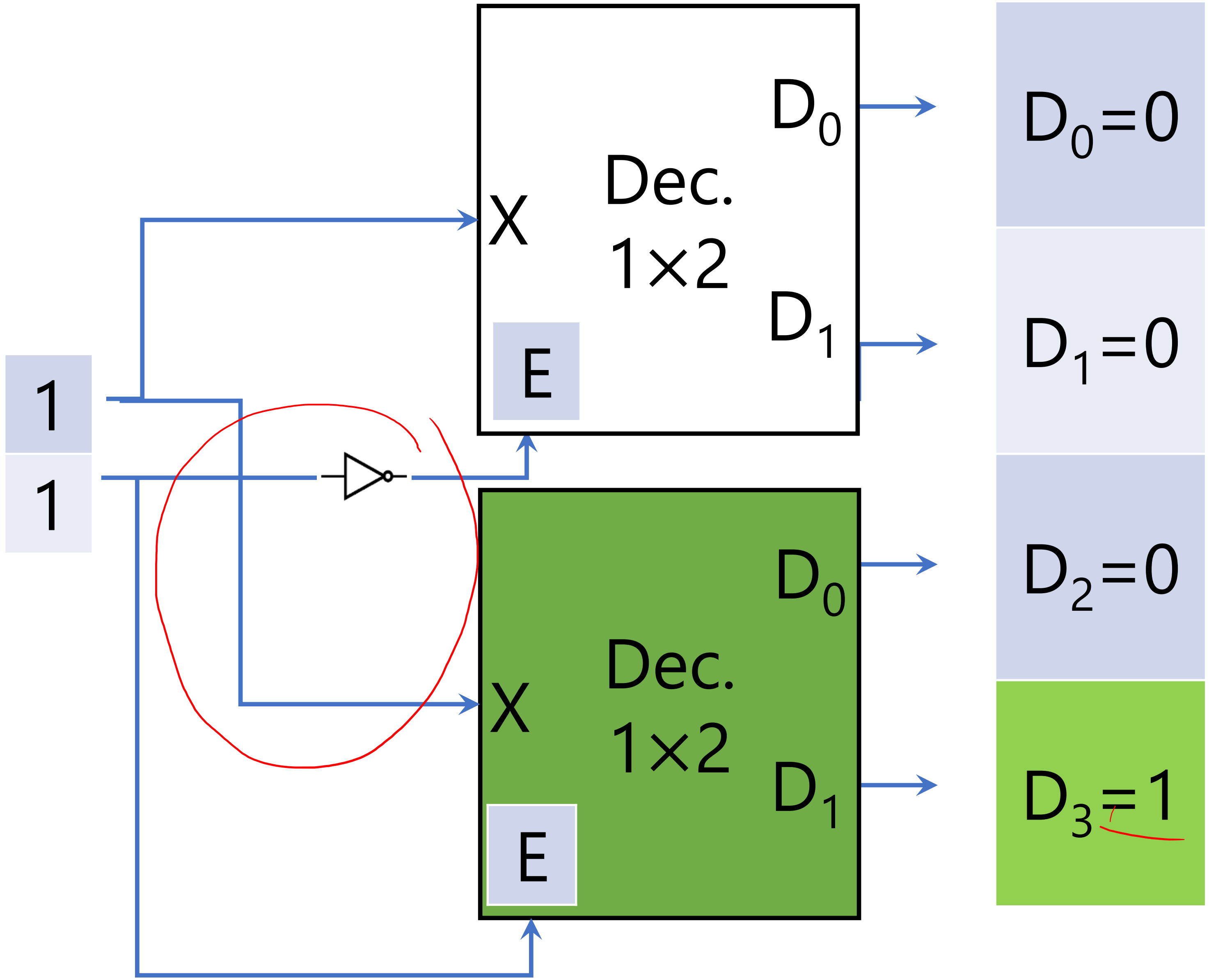


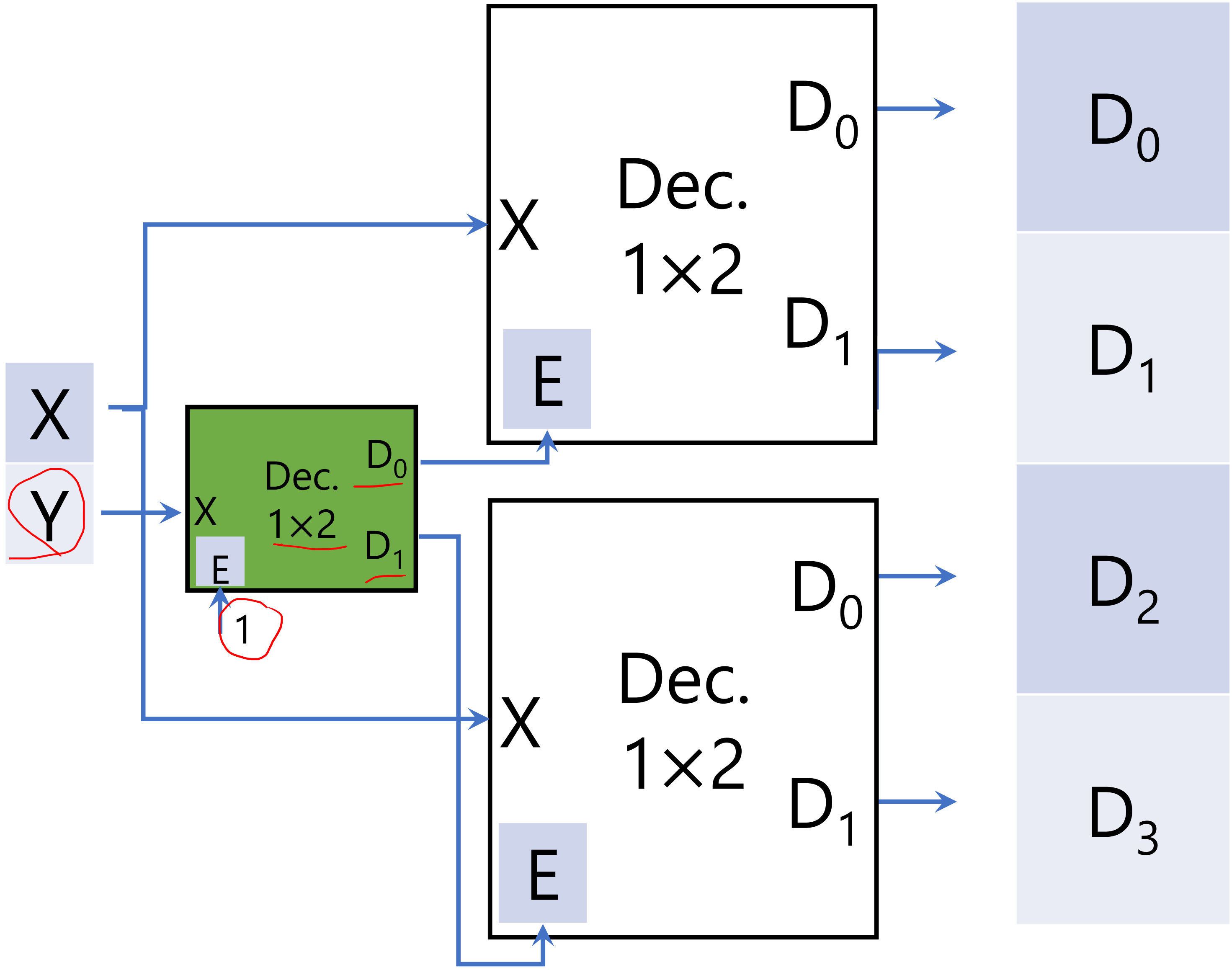
1
0
X
Dec.
1×2
D0
D1
E
X
Dec.
1×2
D0
D1
E
D0=0
D1=1
D2=0
D3=0

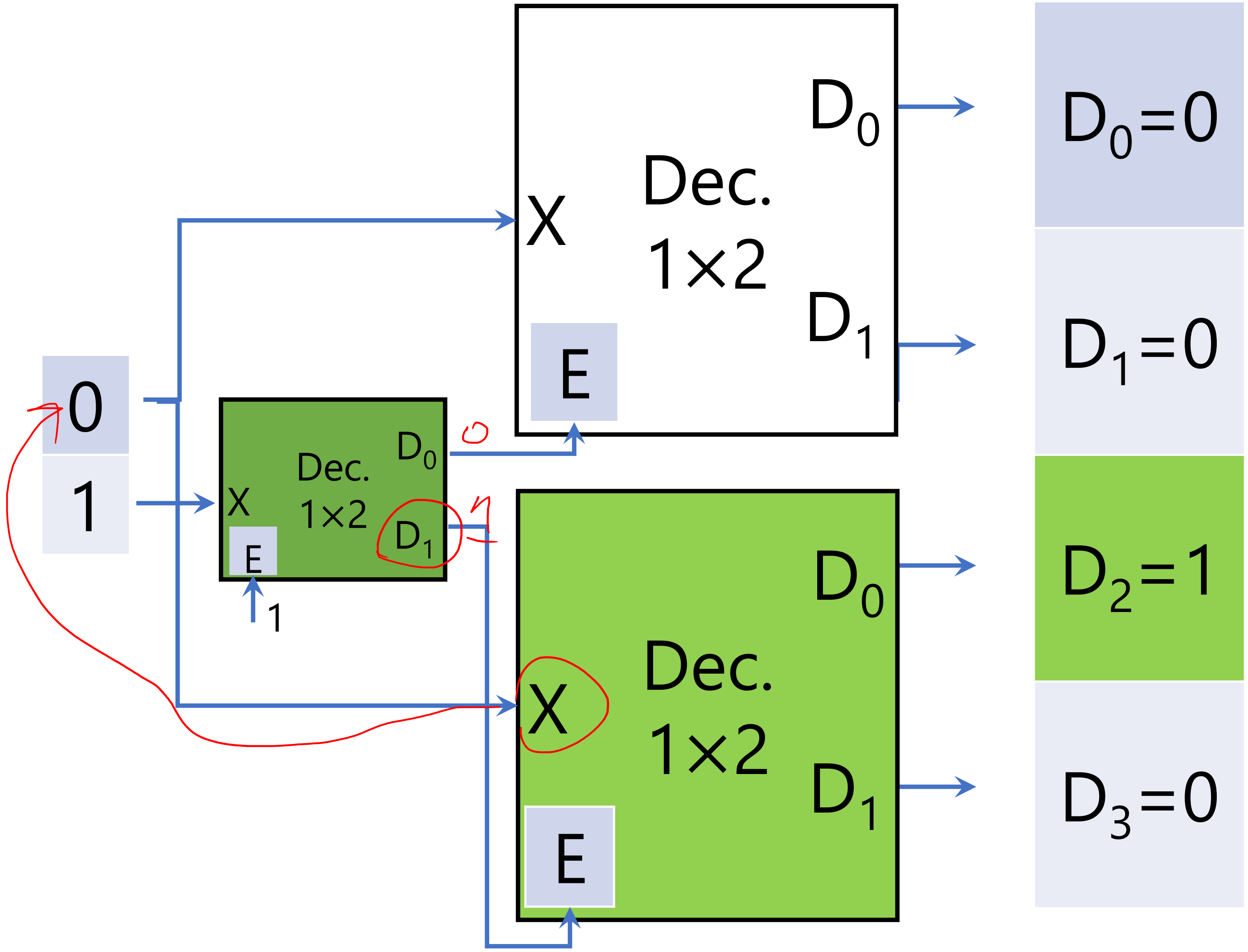

X
1
0
X
Dec.
1×2
E
D_0
D_1
X
Dec.
1×2
E
D_0
D_1
D_0
D_1
D_2
D_3



0
1
X
Dec.
1×2
E
D_0
D_1
X
Dec.
1×2
E
D_0
D_1
$D_0=0$
$D_1=0$
$D_2=1$
$D_3=0$


1
1
X
Dec.
1×2
D0
D1
E
X
Dec.
1×2
D0
D1
E
D0=0
D1=0
D2=0
D3=1


X
Y
Dec.
1×2
X
E
D0
D1
1
Dec.
1×2
X
E
D0
D1
Dec.
1×2
X
E
D0
D1
D0
D1
D2
D3

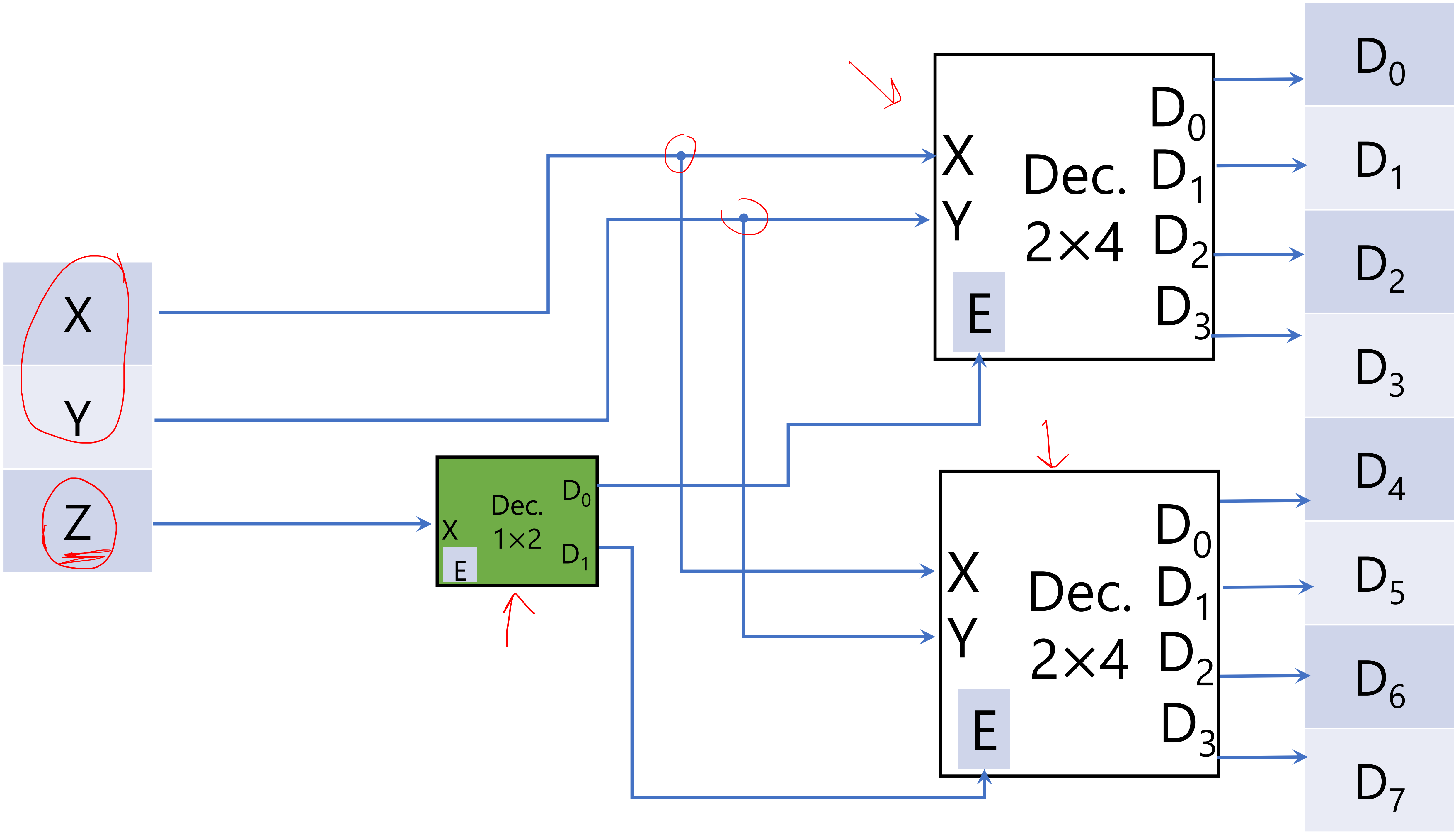


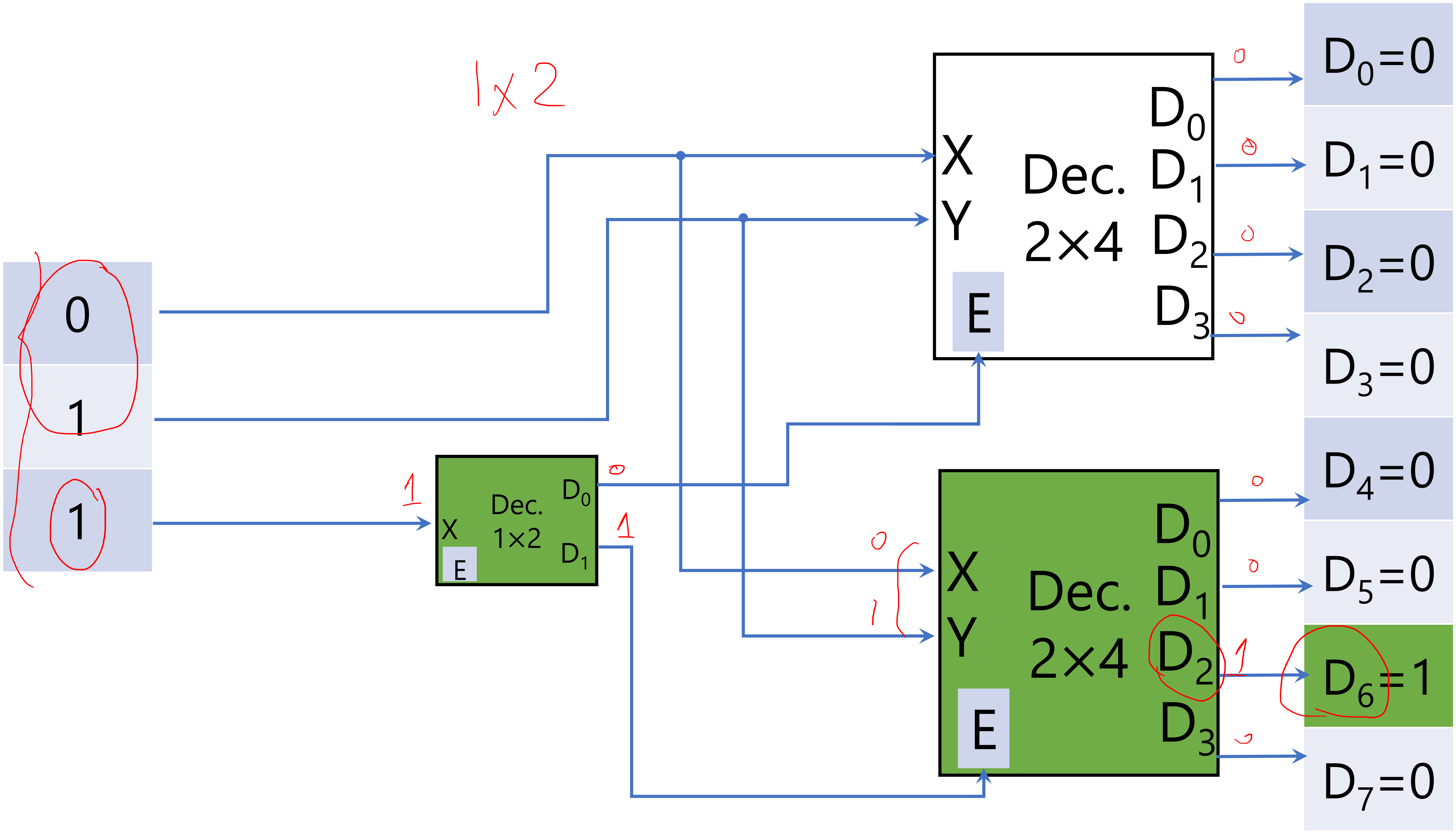
0
1
X
Dec.
1×2
D0
D1
E
1
0
1
X
Dec.
1×2
D0
D1
E
X
Dec.
1×2
D0
D1
E
D0=0
D1=0
D2=1
D3=0

Decoder

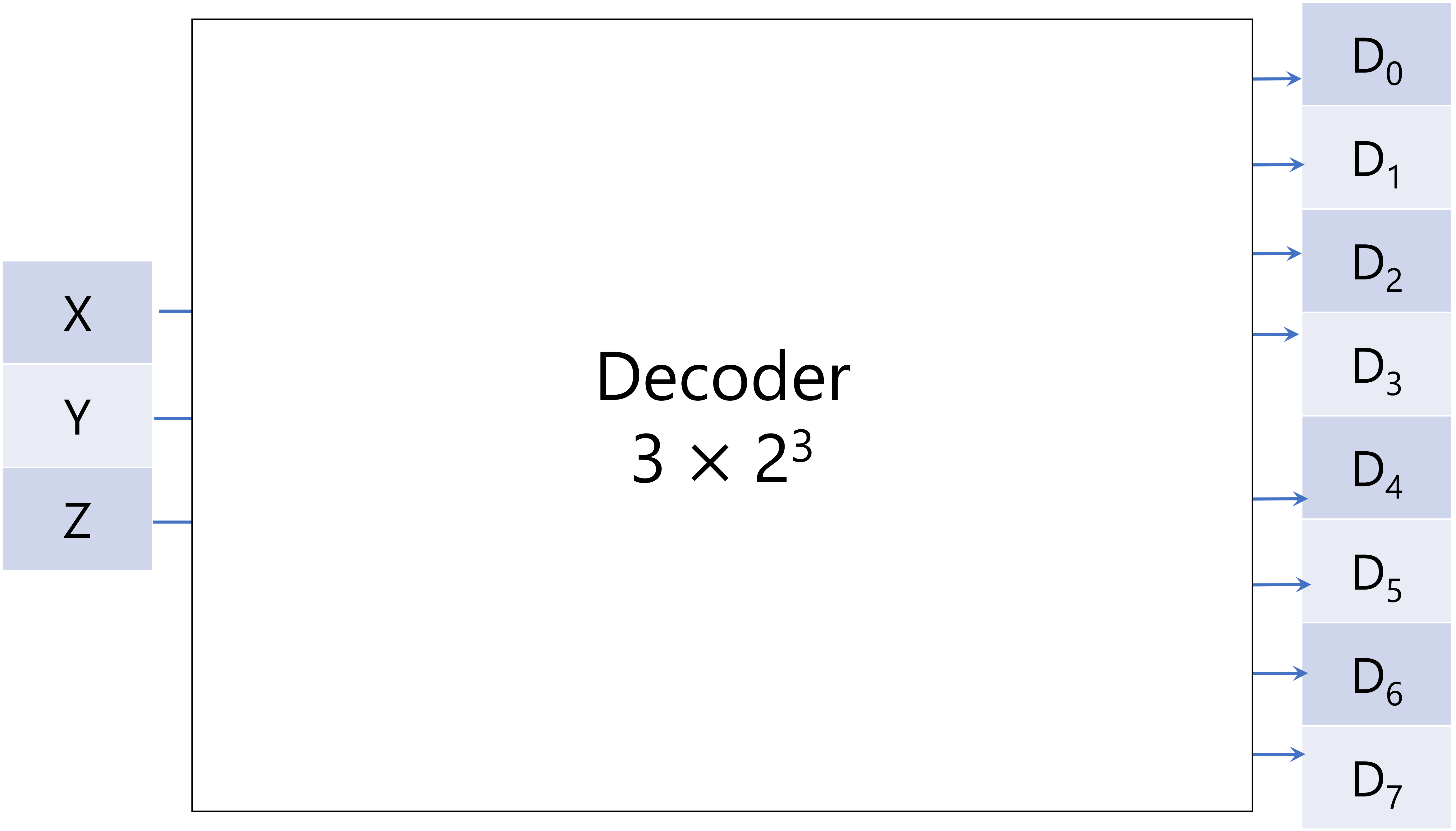
Decode 3-Bit Binary to 2^3 One-hot

Re-Use 2×2^2 Decoder


X
Y
Z
X
Dec.
1×2
D0
D1
E
X
Y
Dec.
2×4
D0
D1
D2
D3
E
X
Y
Dec.
2×4
D0
D1
D2
D3
E
D0
D1
D2
D3
D4
D5
D6
D7



1x2
0
1
1
X
Dec.
1×2
E
D0
D1
X
Y
Dec.
2×4
E
D0
D1
D2
D3
X
Y
Dec.
2×4
E
D0
D1
D2
D3
D0=0
D1=0
D2=0
D3=0
D4=0
D5=0
D6=1
D7=0


X
Y
Z
Decoder
3×2^3
D_0
D_1
D_2
D_3
D_4
D_5
D_6
D_7

Decoder

Decode 4-Bit Binary to 2^4 One-hot

Re-Use 1×2^1 Decoder
Re-Use 2×2^2 Decoder
Re-Use 3×2^3 Decoder

ORDER STATUS | BOM MANAGER | ABOUT US | HELP

CANADA ENGLISH CAD

All Products

Login or REGISTER

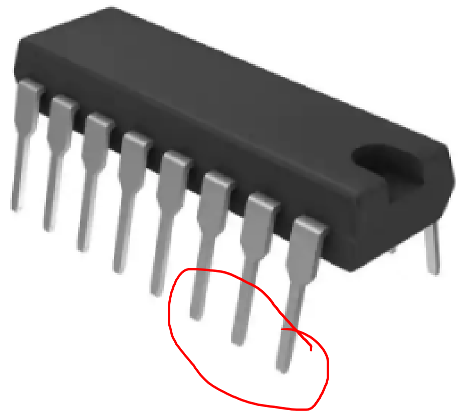
0 ITEM(S)

Products Manufacturers Resources Tools

16×2^{16}

Product Index > Integrated Circuits (ICs) > Logic - Signal Switches, Multiplexers, Decoders > Texas Instruments SN74LS138N

Share



SN74LS138N

Datasheet

Digi-Key Part Number	296-1639-5-ND
Manufacturer	Texas Instruments
Manufacturer Product Number	SN74LS138N
Supplier	Texas Instruments
Description	IC 3-8 LINE DECODER/DEMUX 16-DIP
Manufacturer Standard Lead Time	6 Weeks
Detailed Description	Decoder/Demultiplexer 1 x 3:8 16-PDIP

Customer Reference

Media & Downloads

RESOURCE TYPE	LINK
Datasheets	SN54LS138, SN54S138, SN74LS138, SN74S138A
Featured Product	Logic Solutions Analog Solutions
PCN Design/Specification	Material Set 30/Mar/2017
EDA / CAD Models	SN74LS138N by SnapEDA SN74LS138N by Ultra Librarian

Price and Procurement

4,043 In Stock

Can ship immediately

QUANTITY

Quantity

Add to Cart

Add to BOM

Add to Favorites

Tube

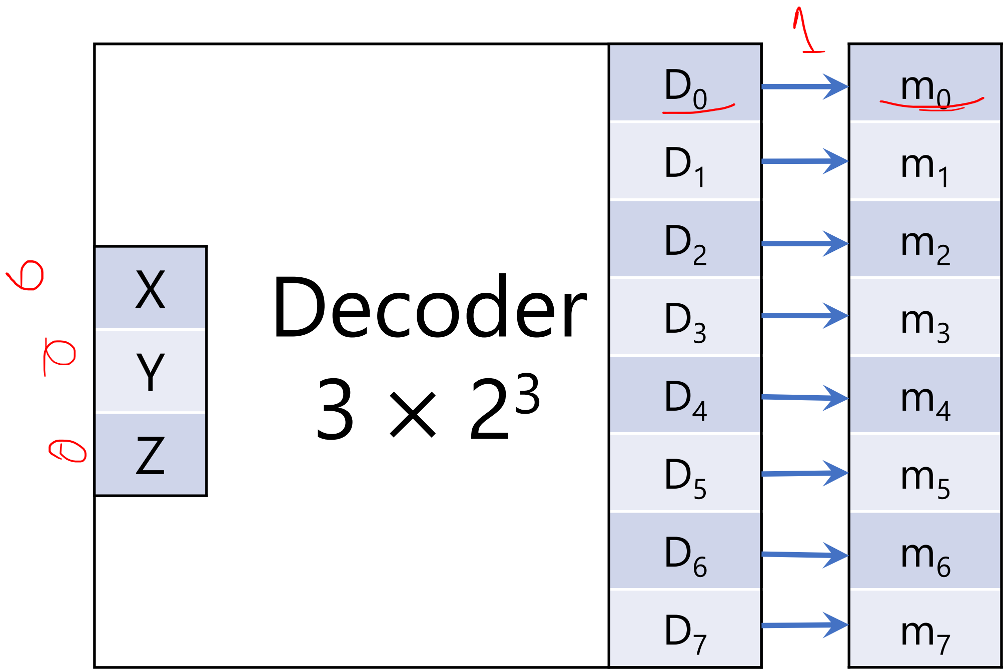
QTY	UNIT PRICE	EXT PRICE
1	$1.27000	$1.27
10	$1.12000	$11.20
25	$1.05280	$26.32
100	$0.85920	$85.92

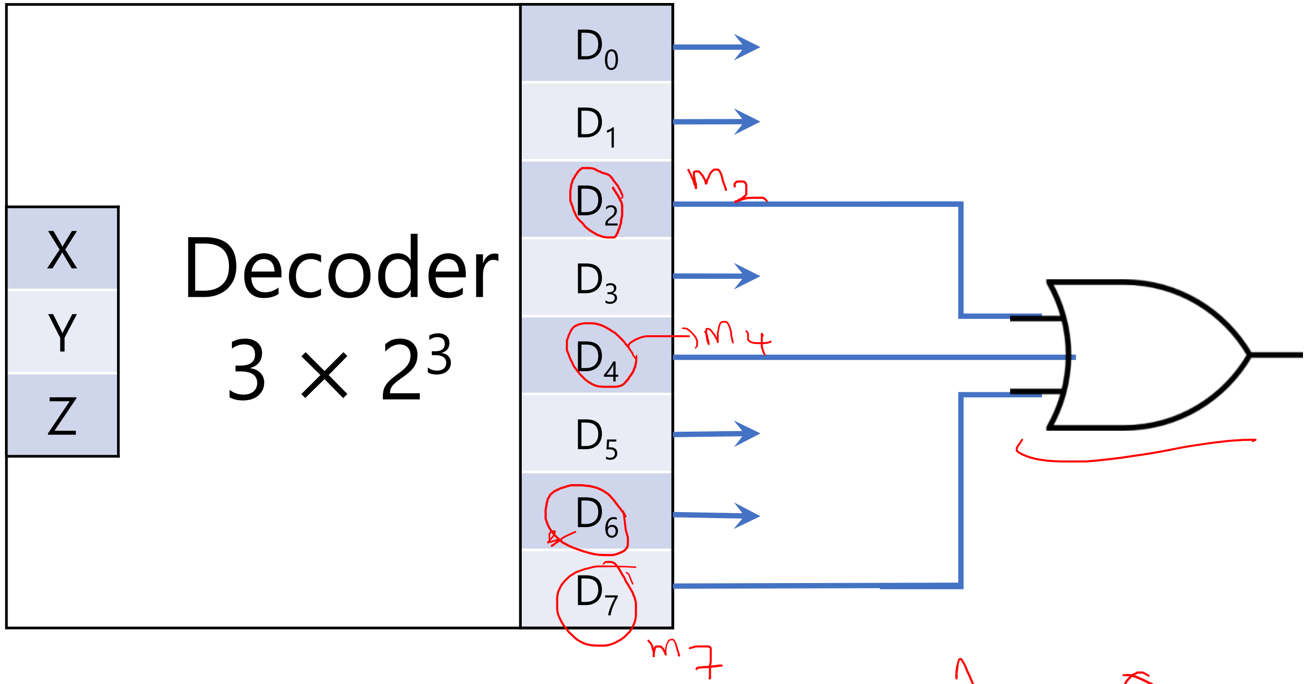
Decoder

Boolean Function

$$F_{SoP} = \sum m(\ldots)$$

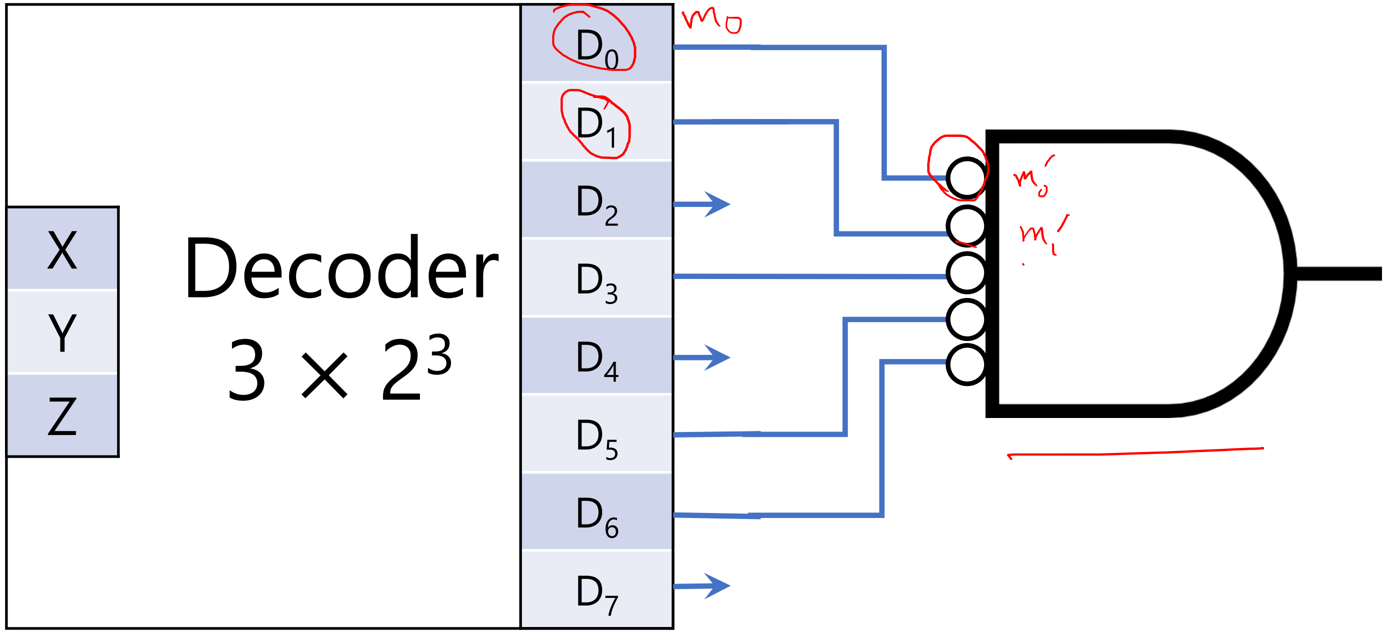
$$F_{PoS} = \prod M(\ldots)$$


X
Y
Z
Decoder
3 × 2³
D0
D1
D2
D3
D4
D5
D6
D7
m0
m1
m2
m3
m4
m5
m6
m7



$$F_{SoP} = \sum m(2,4,7)$$

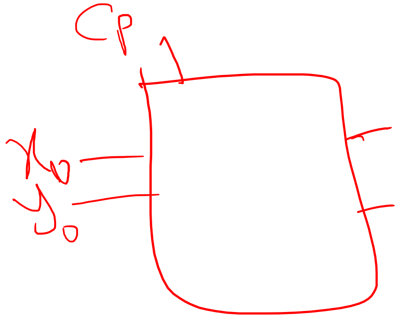
$$1 = m_2 + m_4 + m_7$$



$$F_{PoS} = \prod M(0,1,3,5,6) = \prod m_0' m_1' m_3' m_5' m_6'$$

(m_0') $M_0 M_1 M_3 M_5 M_6$

m_1



$S = x_0 \oplus y_0 \oplus C_p$

$C = x_0 y_0 + C_p x_0 + C_p y_0$

Decoder
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

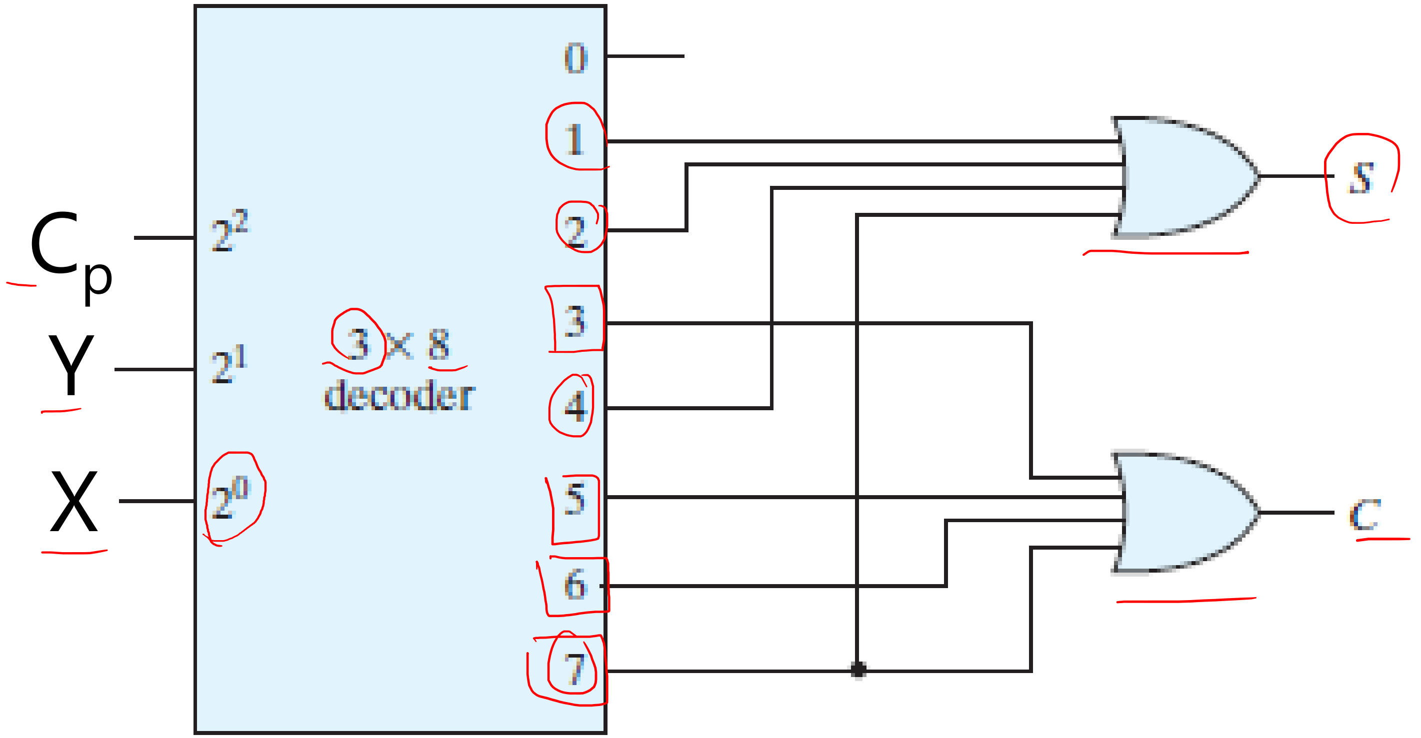


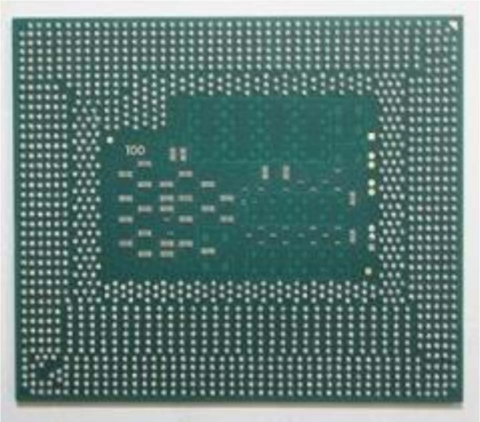
FIGURE 4.21
Implementation of a full adder with a decoder

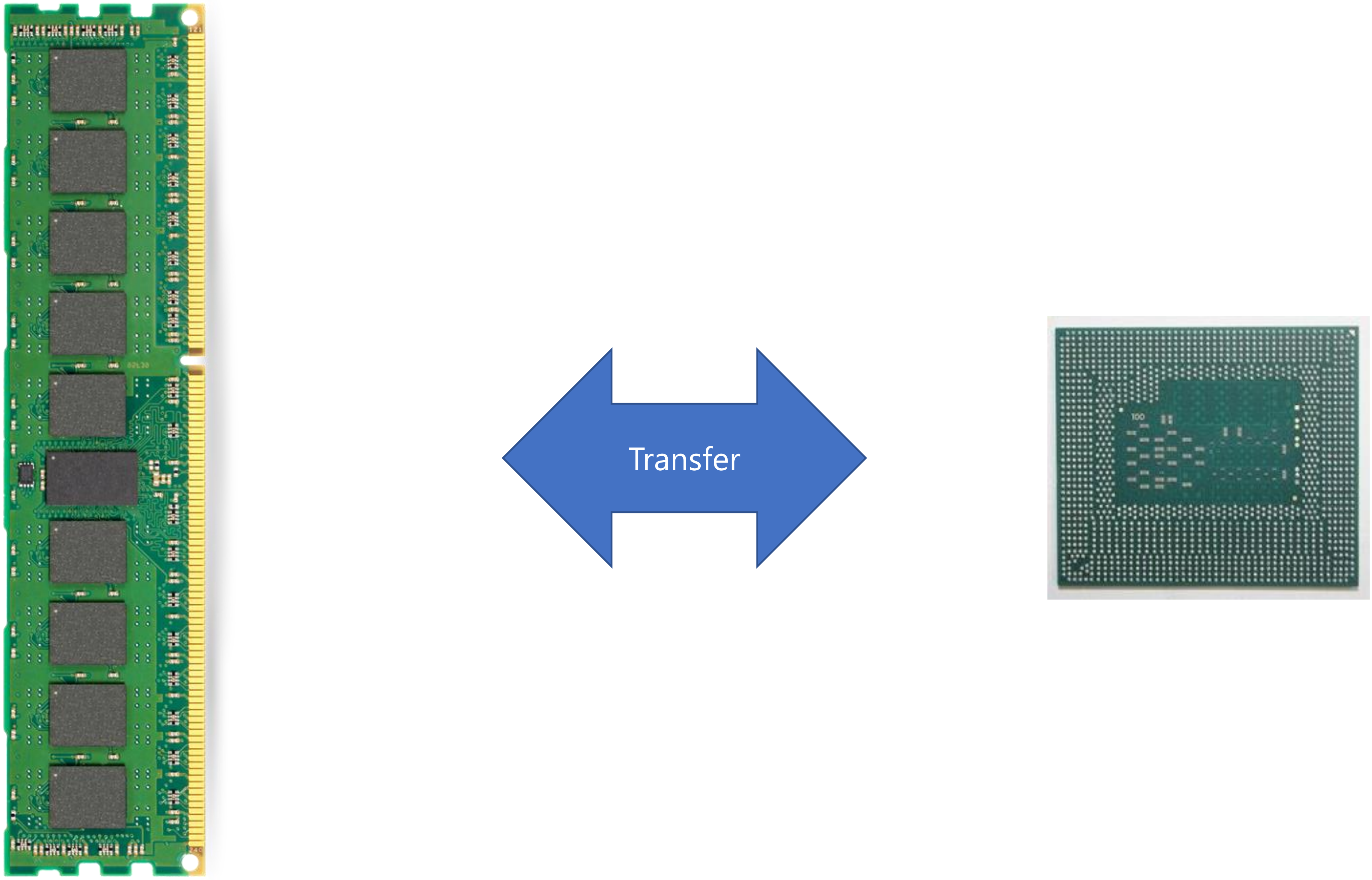

Data
Transmission
Decoder, Encoder
Multiplexer (MUX, MPX), De-Multiplexer (Demux)

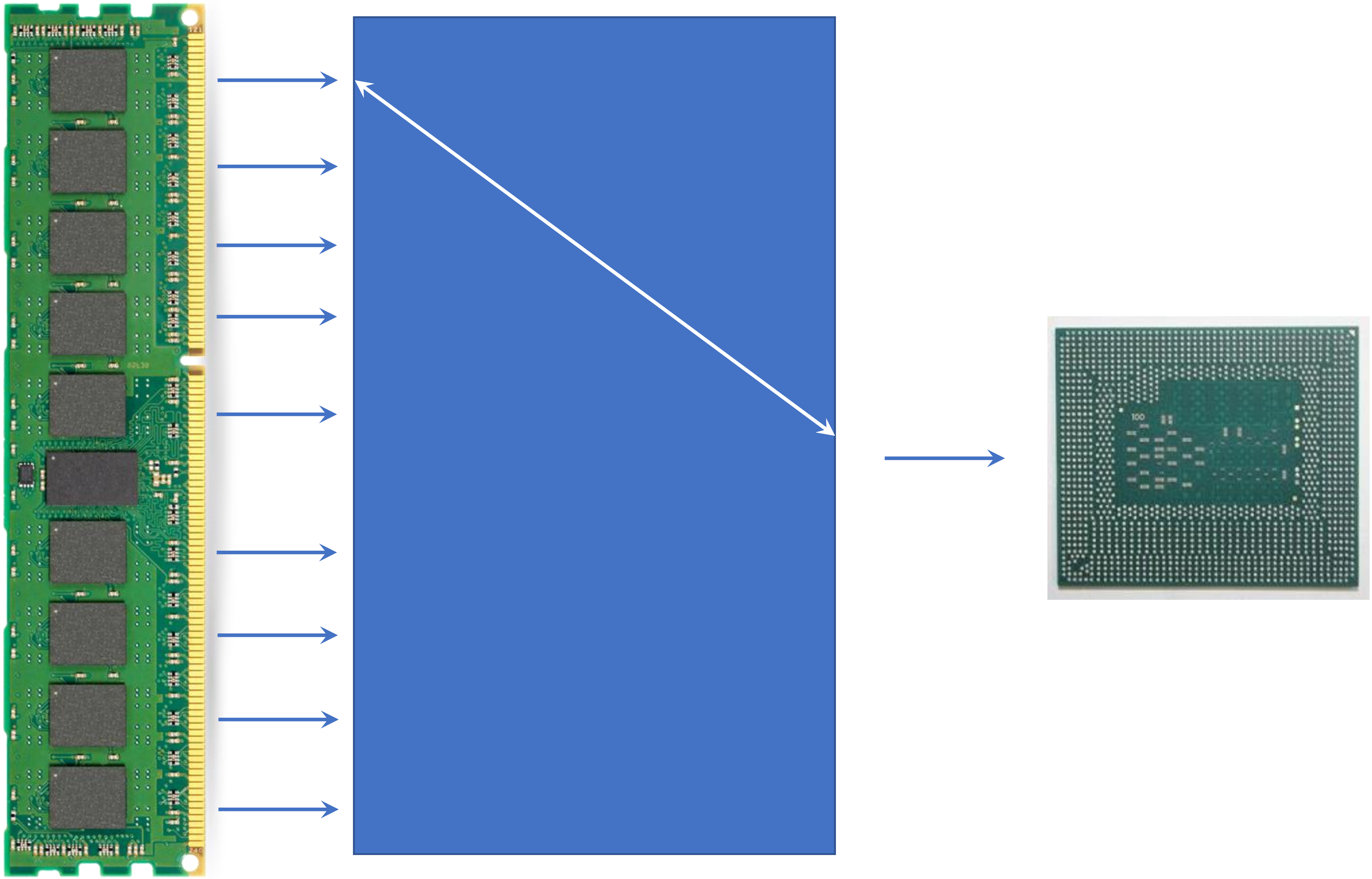
Multiplexer

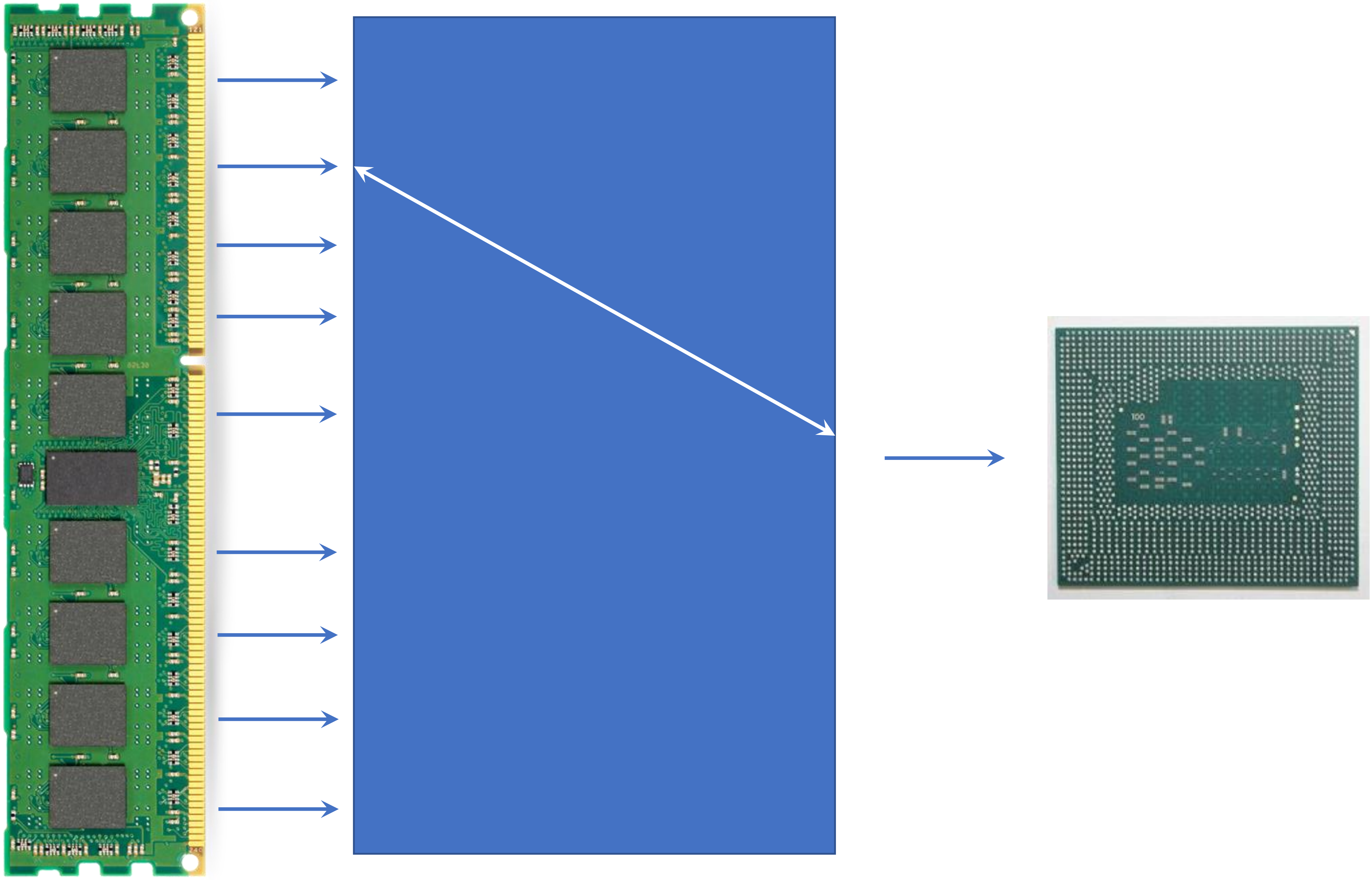
Shortened to MUX or MPX

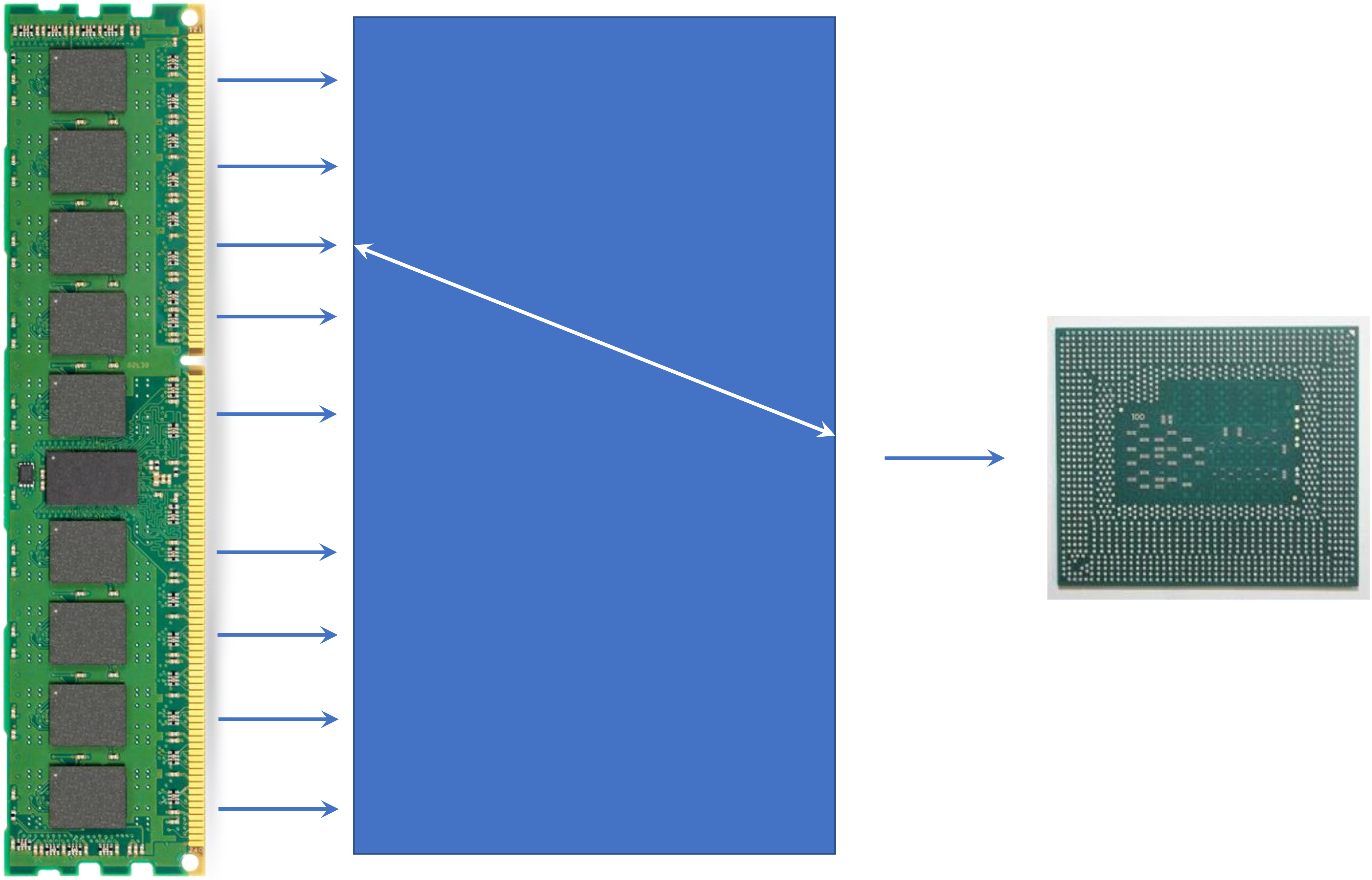
Address to Select

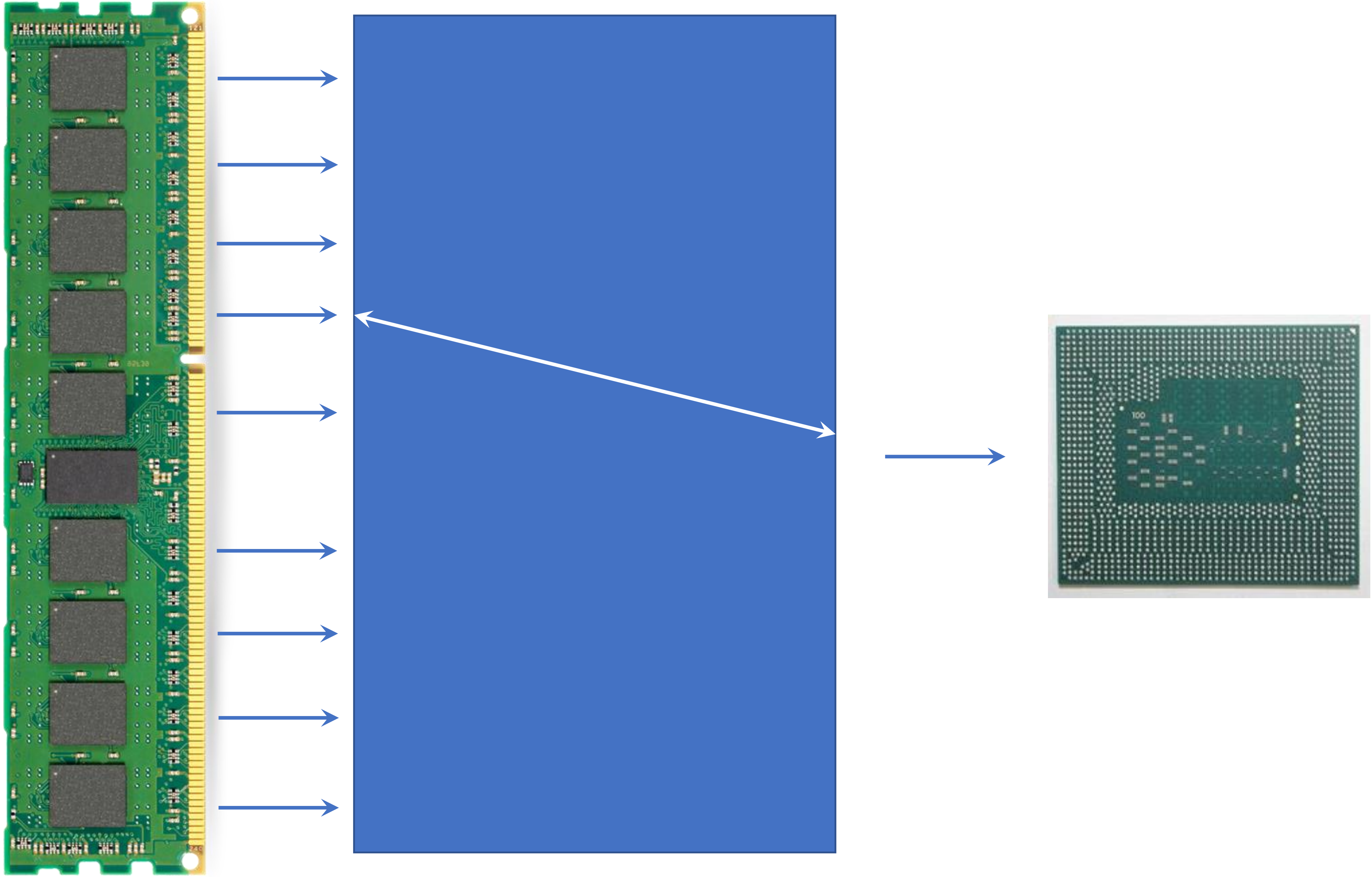


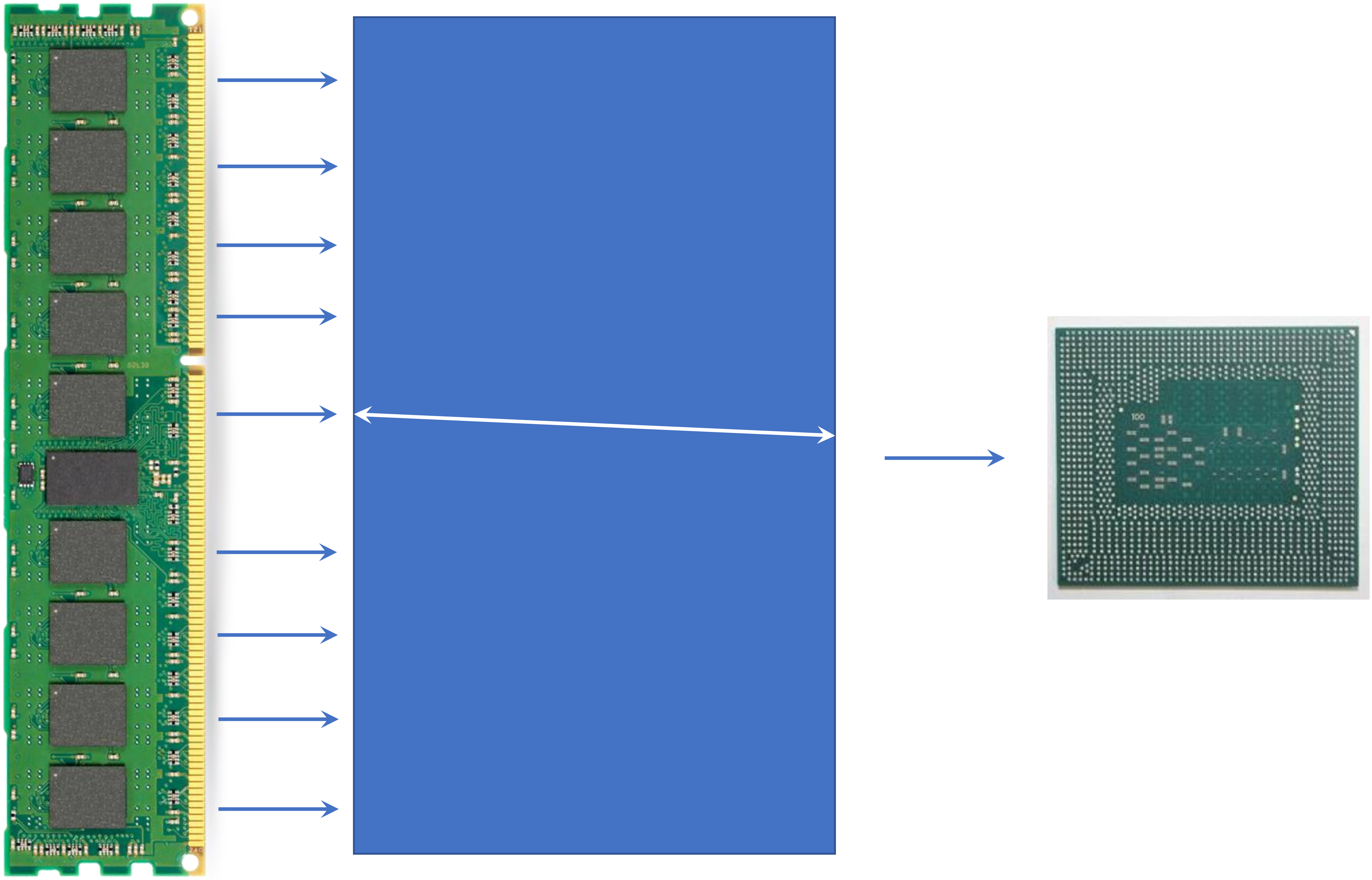

Transfer

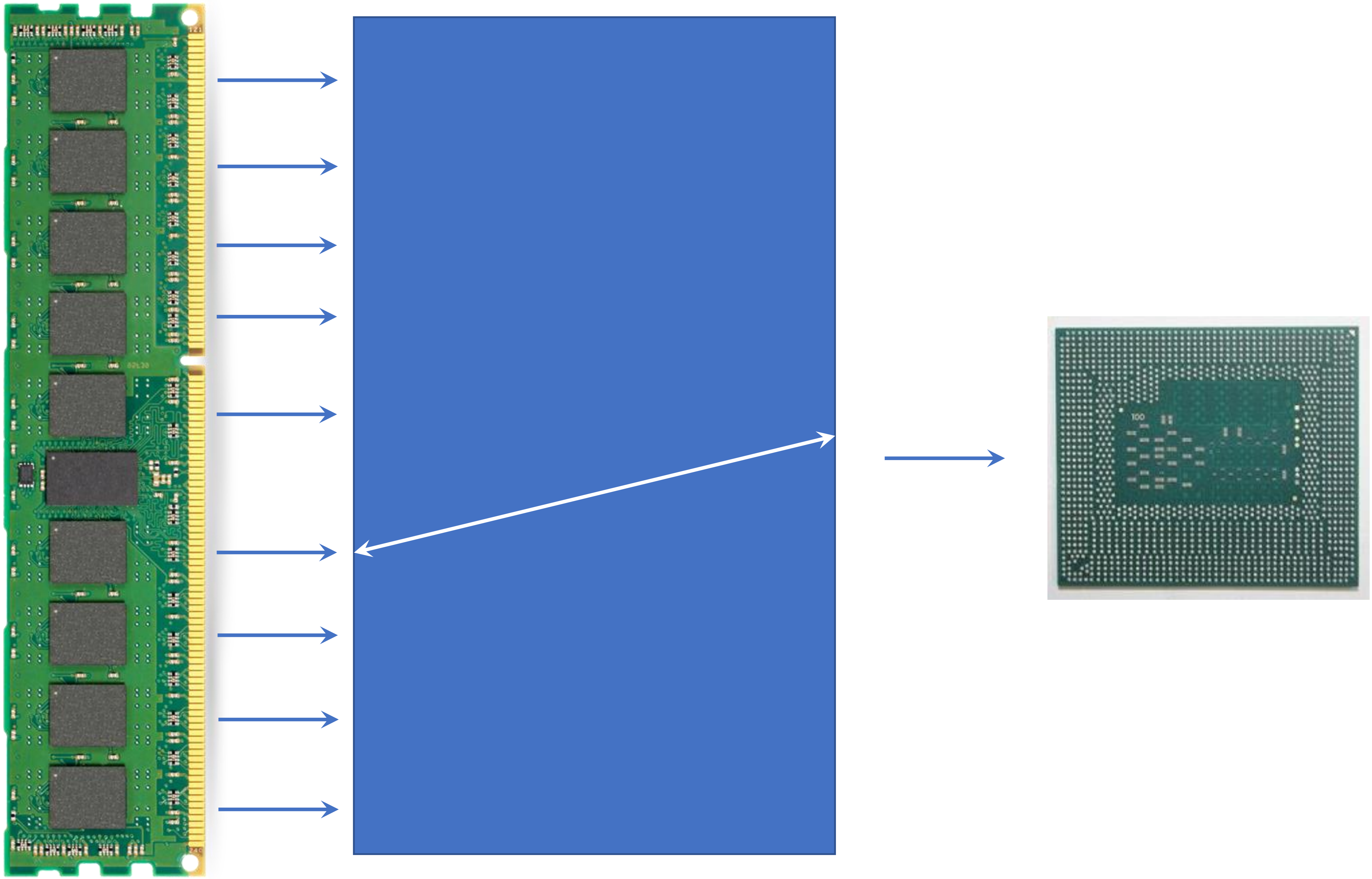


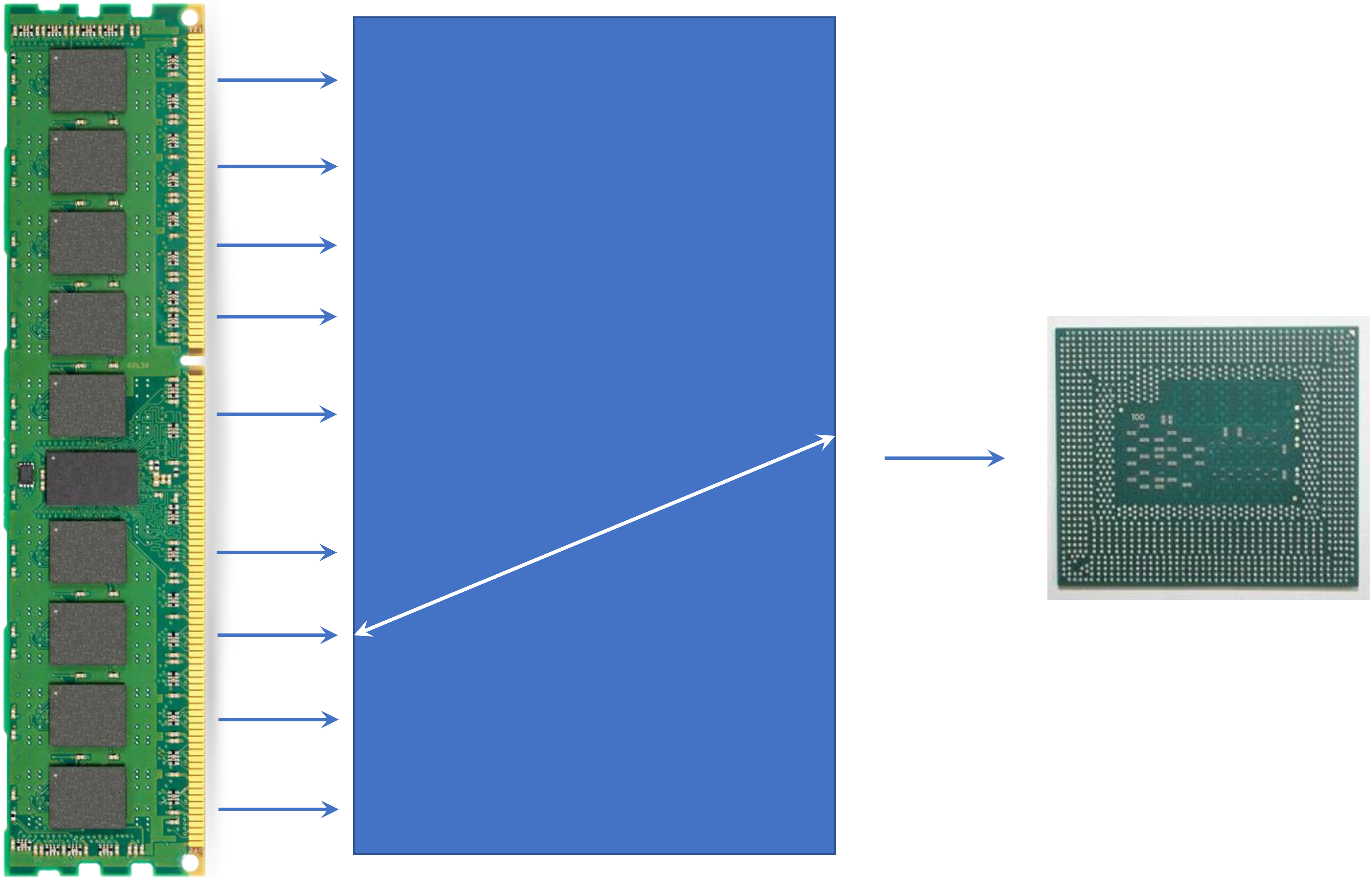


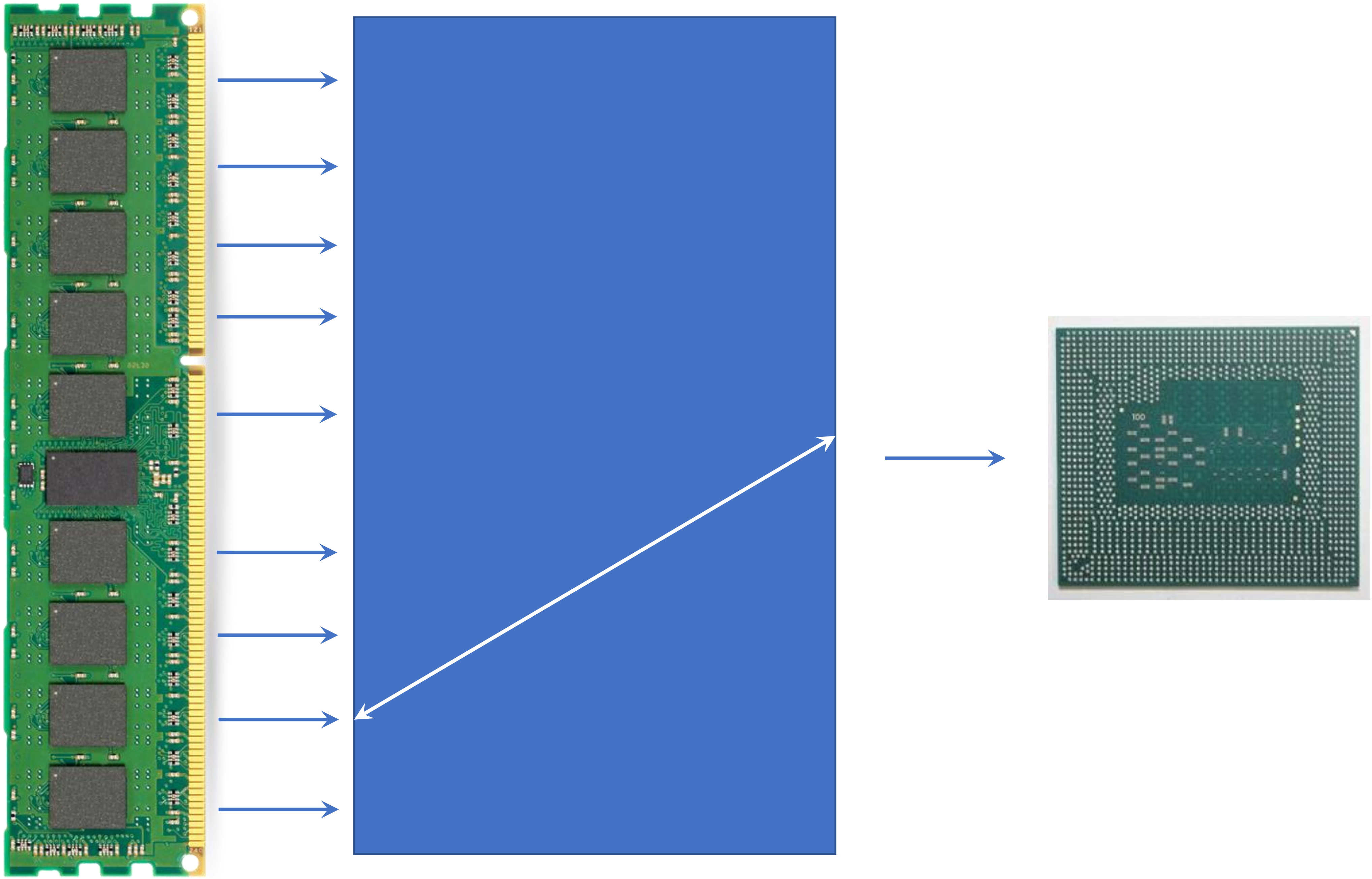


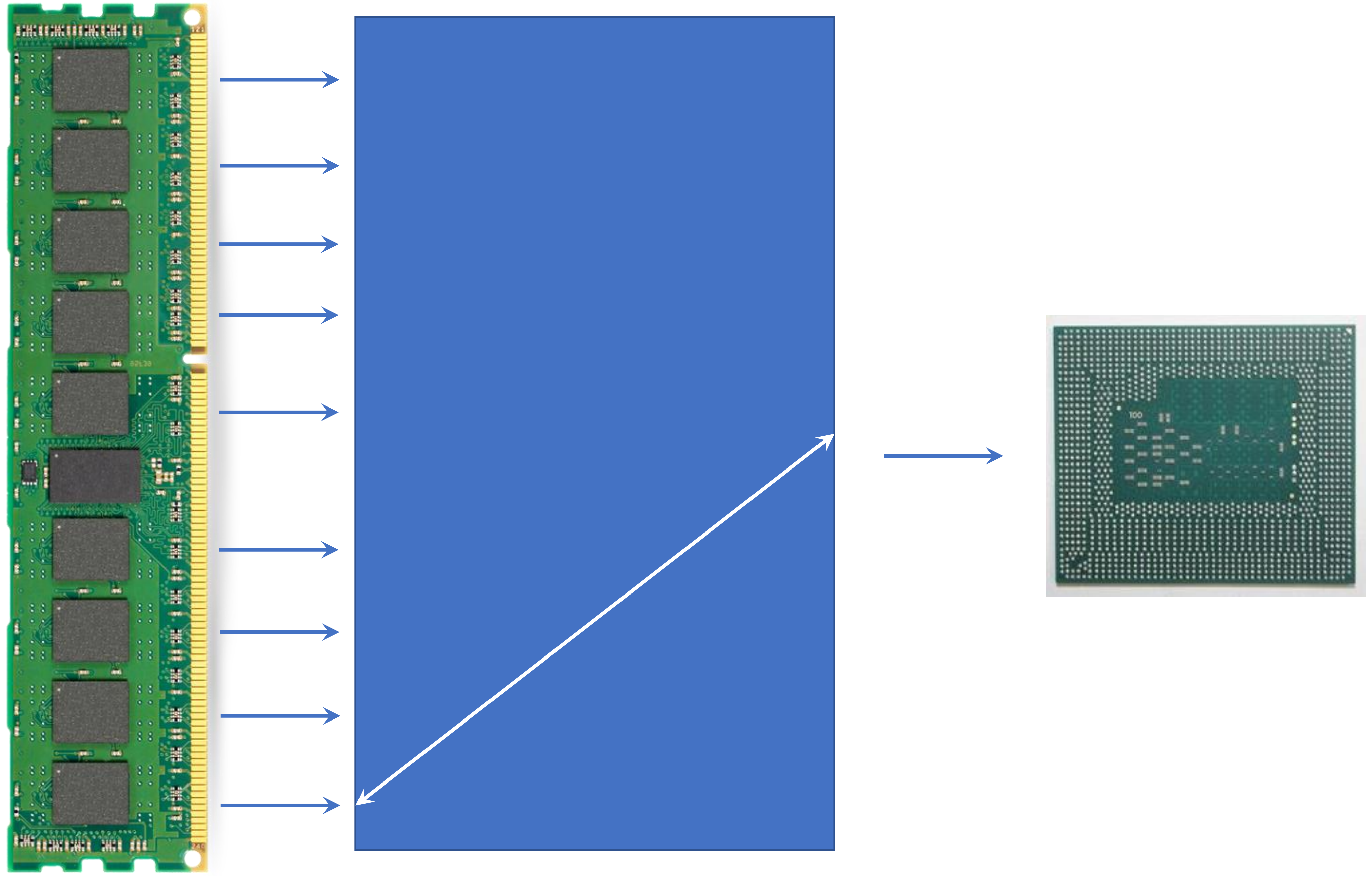




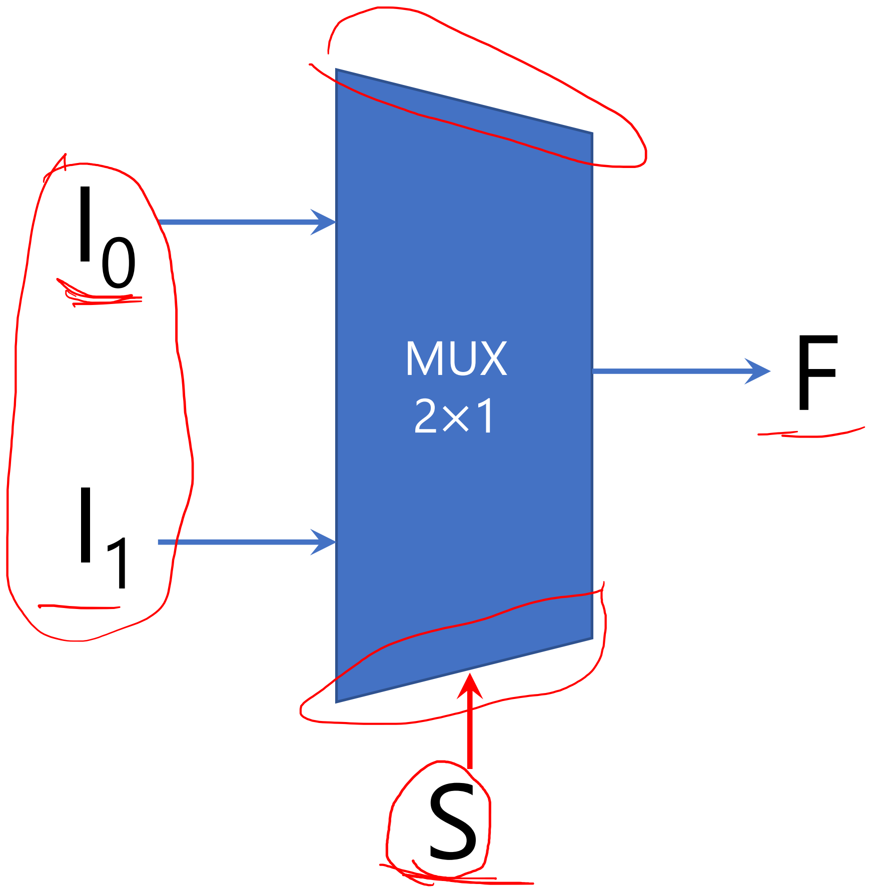


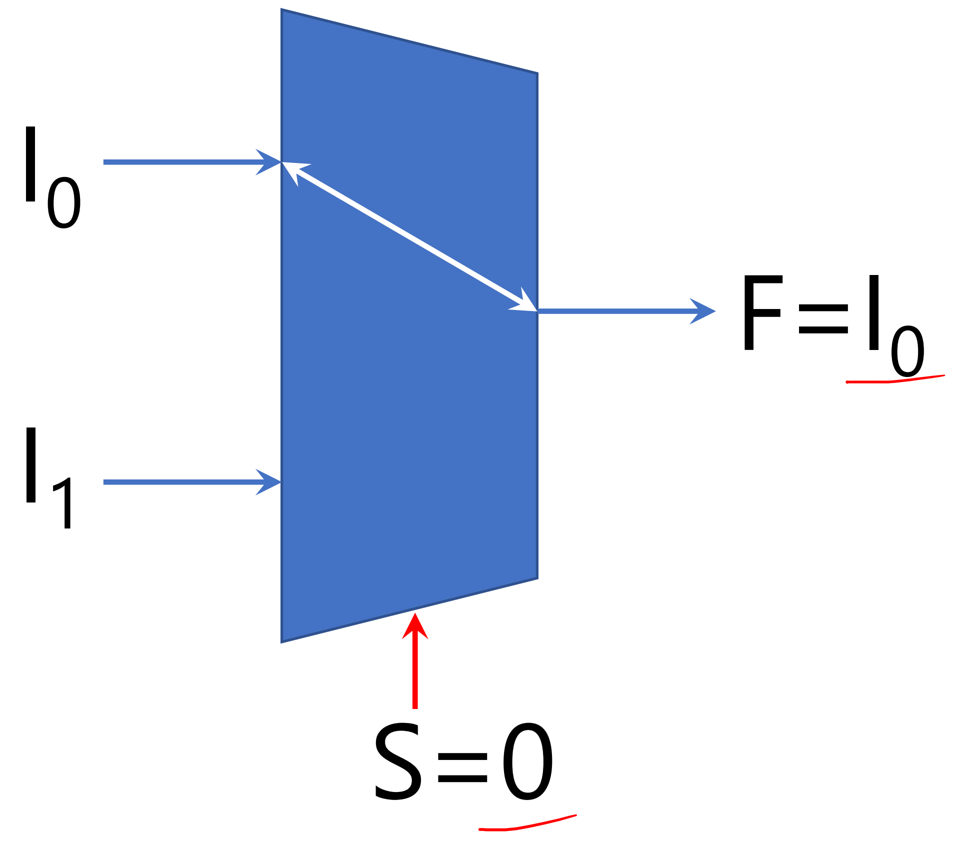


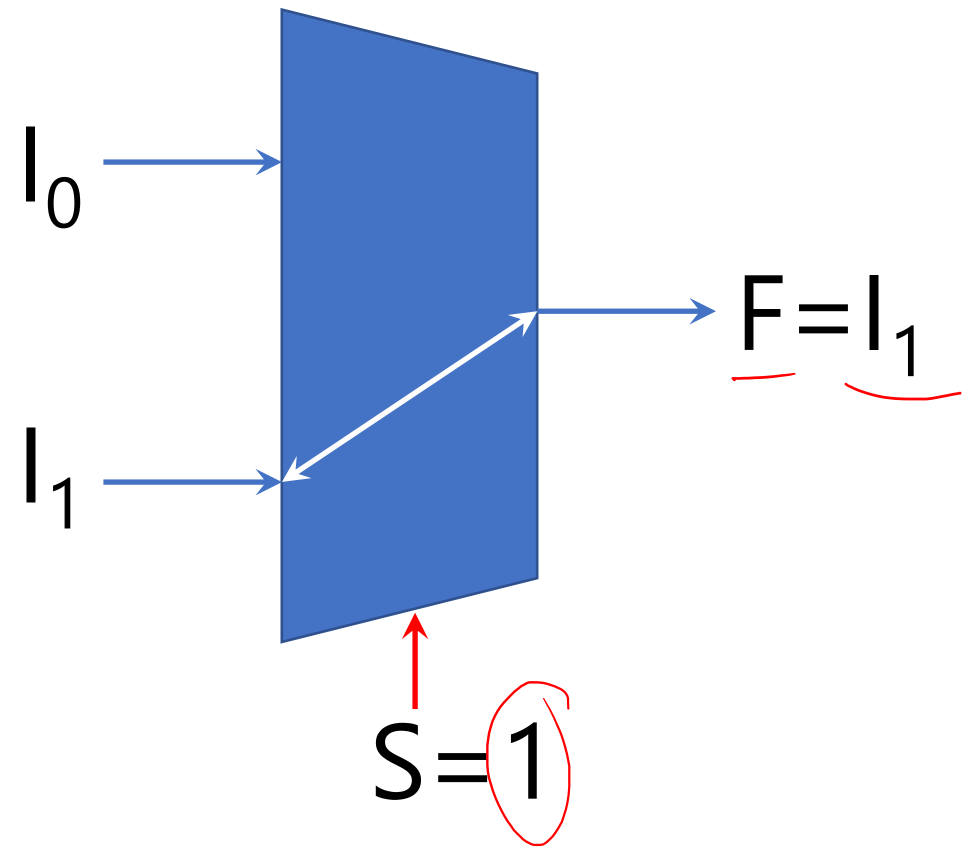




Multiplexer
$2^1 \times 1$


I0
I1
MUX
2×1
F
S


I_0
I_1
$F=I_0$
$S=0$



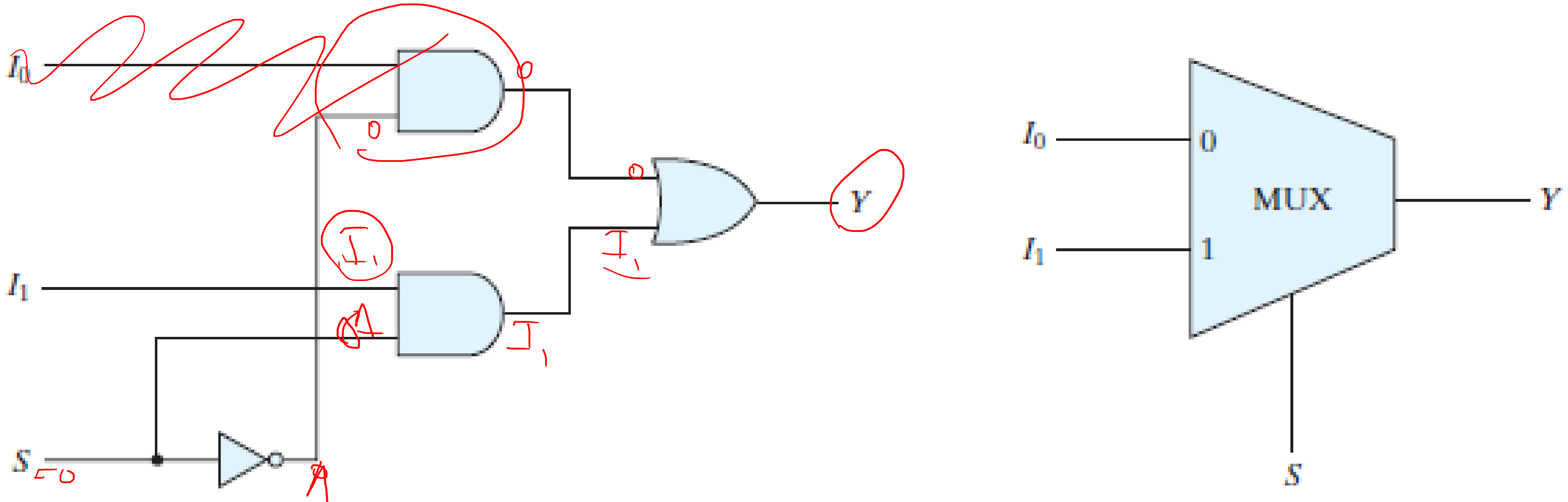
I_0
I_1
$F=I_1$
$S=1$

S	I_1	I_0		F
0	0	0	m_0	0
0	0	1	m_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	m_7	1

S	I_1	I_0	F $= \sum m(1,3,6,7)$
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

$S \backslash I_1I_0$	00	01	11	10
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

$$F = S'I_0 + SI_1$$



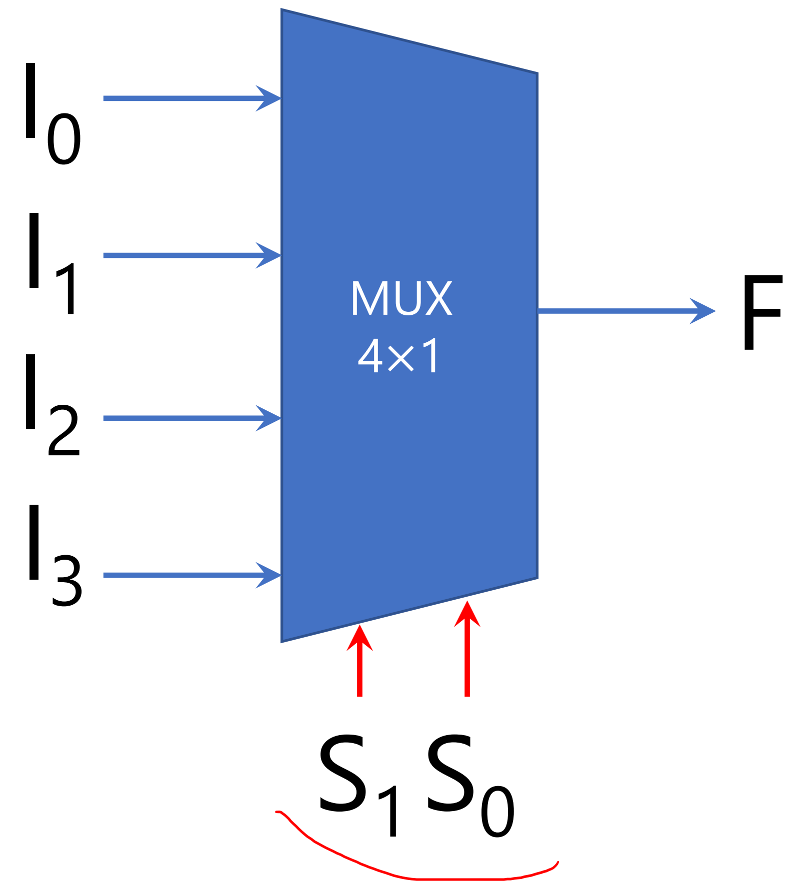
(a) Logic diagram

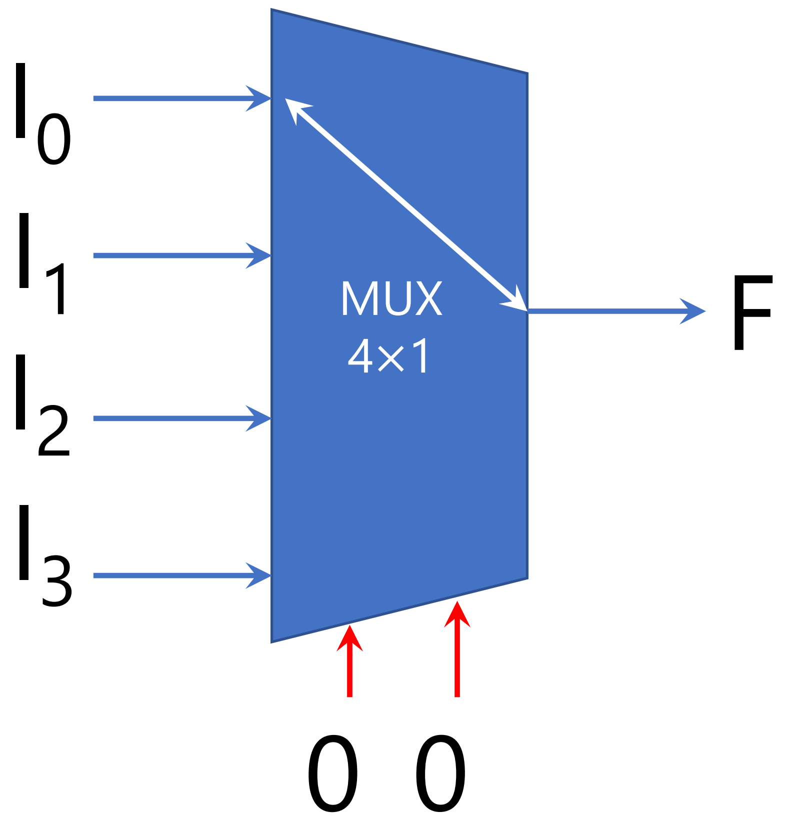
(b) Block diagram

FIGURE 4.24
Two-to-one-line multiplexer

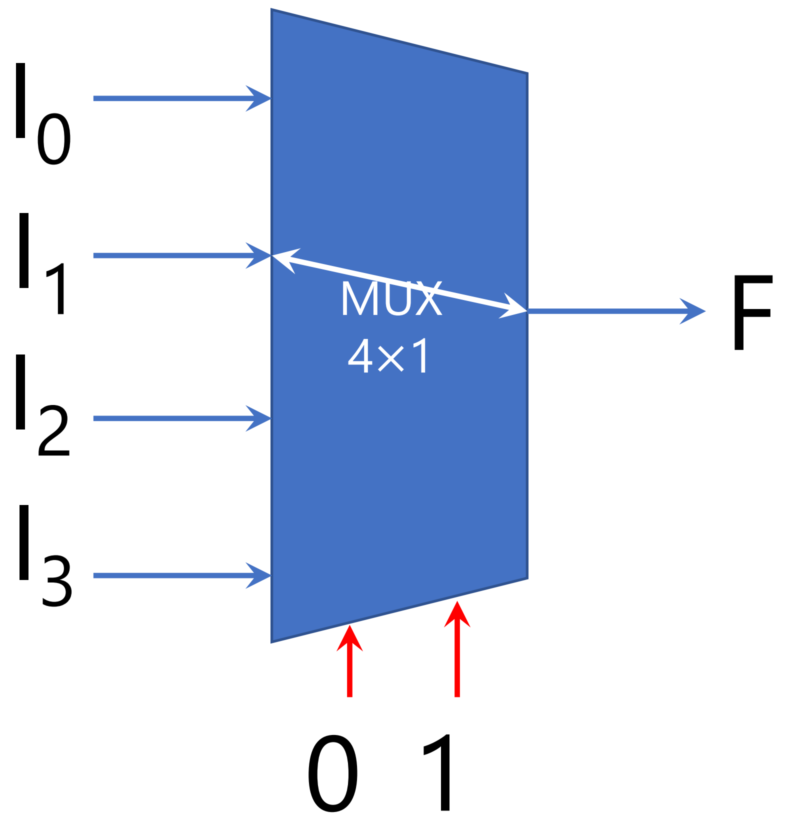
S	$F = S'I_0 + SI_1$
0	I_0
1	I_1

Multiplexer
$2^2 \times 1$

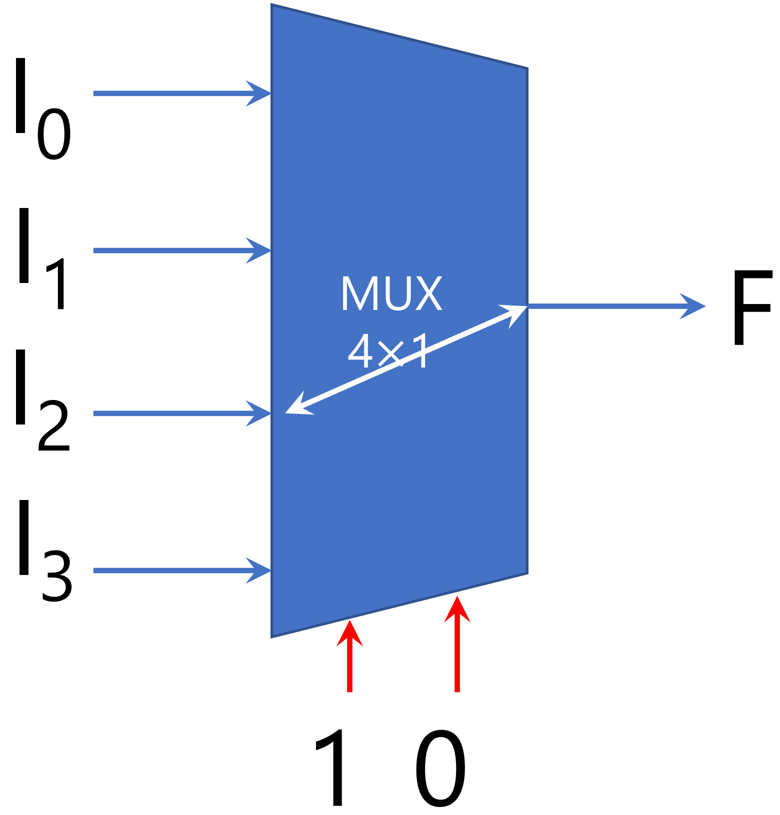

I_0
I_1
I_2
I_3
MUX
4×1
F
S_1 S_0



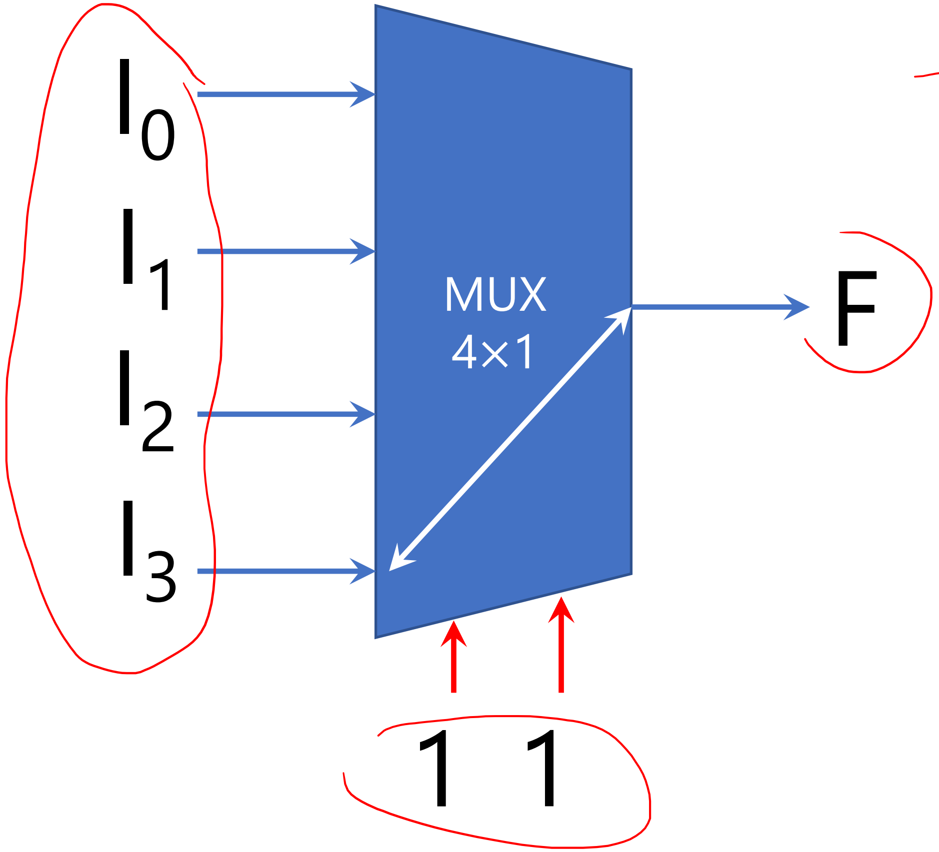
I0
I1
I2
I3
MUX
4×1
F
0 0



I_0
I_1
I_2
I_3
MUX
4×1
F
0 1



I_0
I_1
I_2
I_3
MUX
4×1
F
1 0

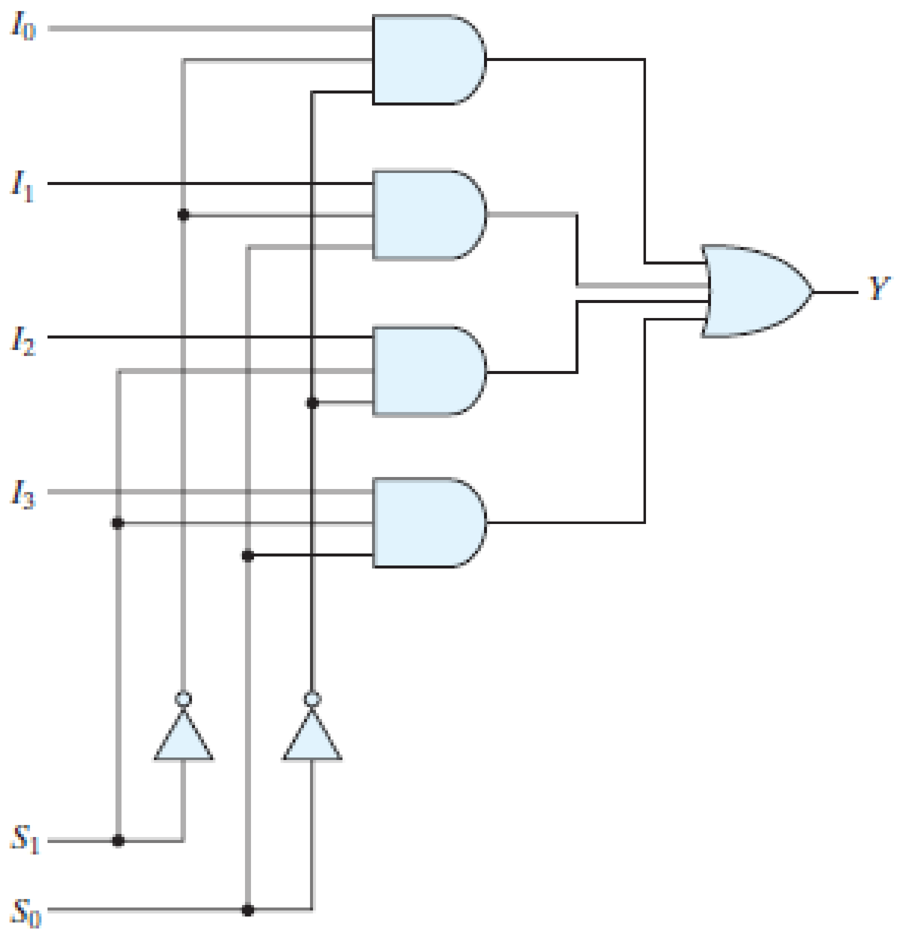


S_1	S_0	I_3	I_2	I_1	I_0	F

→ 2^6

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



(a) Logic diagram

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

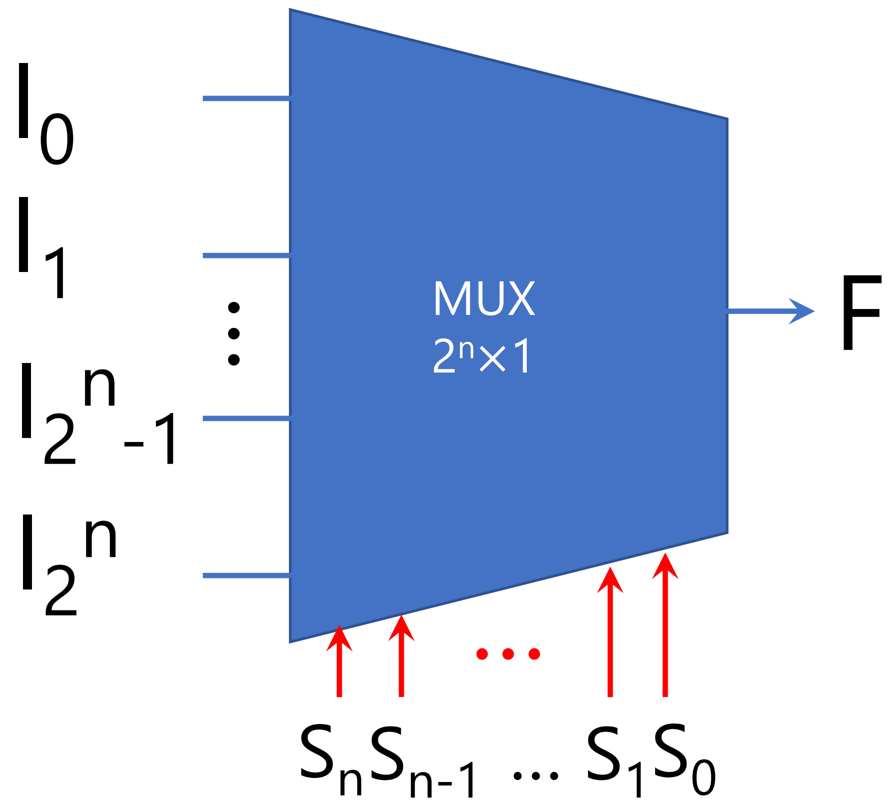
(b) Function table

FIGURE 4.25
Four-to-one-line multiplexer

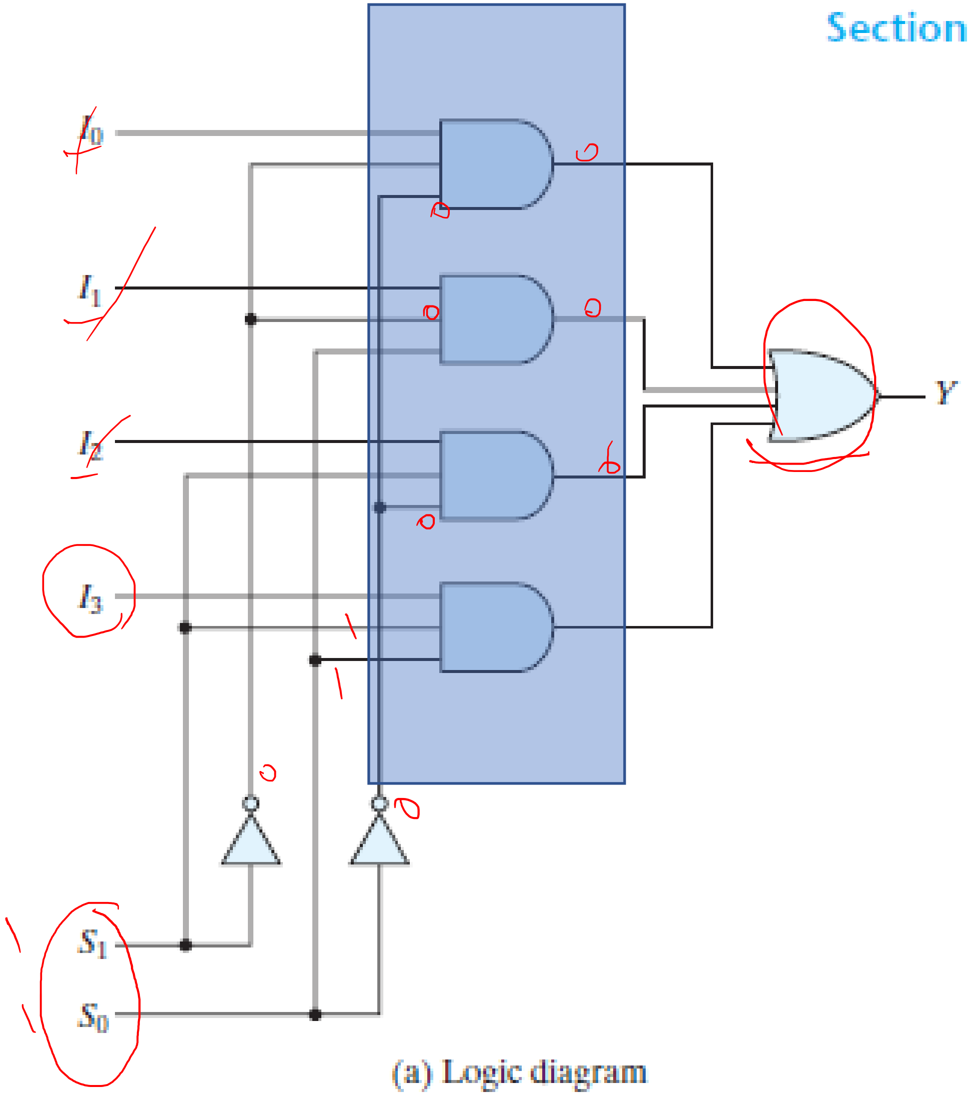
Multiplexer

$2^n \times 1$

$\overbrace{S_{n-1} \cdots S_0}^{n}$



I_0
I_1
I_{2^n-1}
I_{2^n}
MUX
$2^n \times 1$
F
$S_n S_{n-1} \dots S_1 S_0$



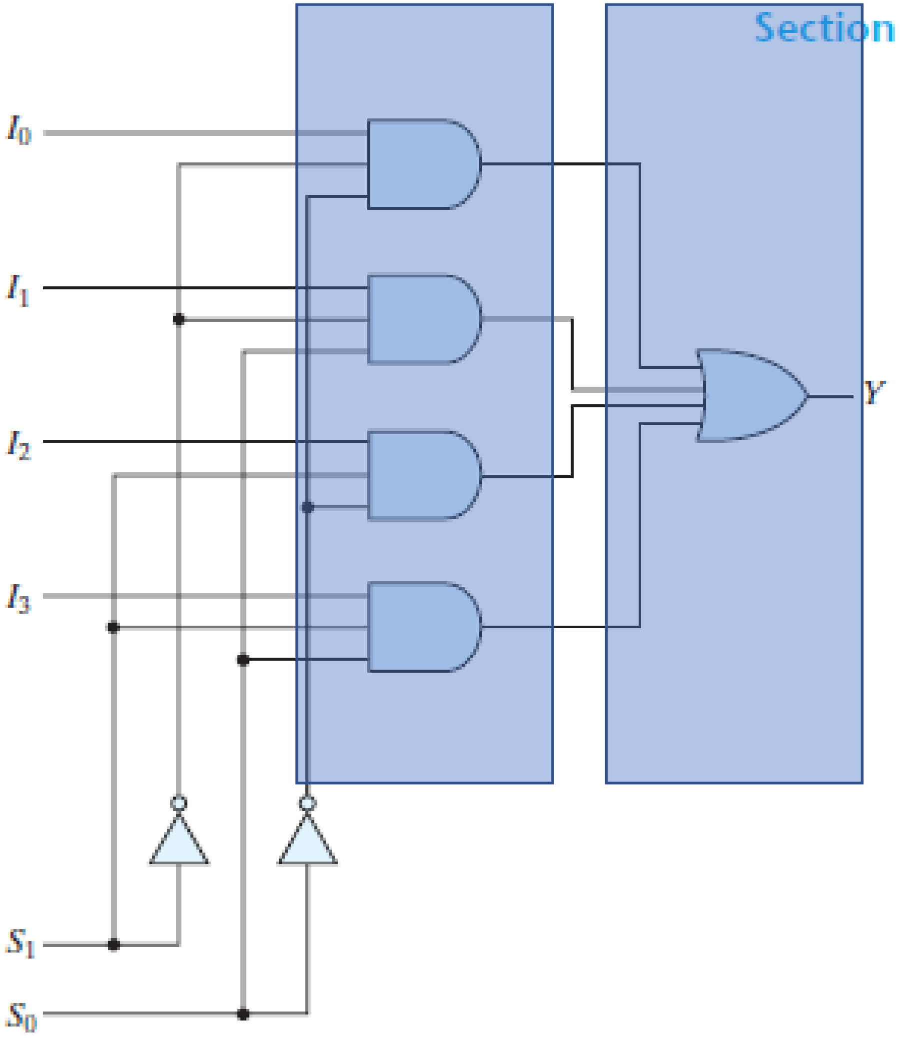
(a) Logic diagram

≈ 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

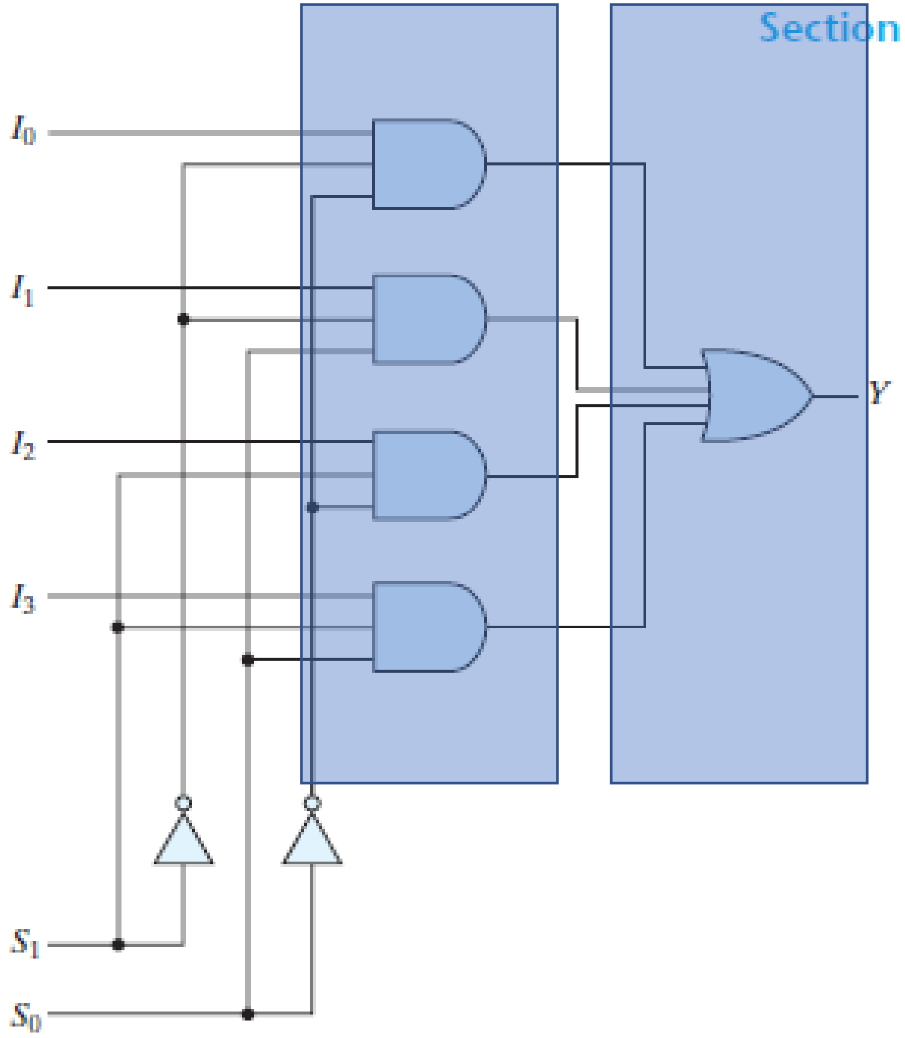


(a) Logic diagram

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



(a) Logic diagram

Sum of Products
2 Levels
ANDs-OR

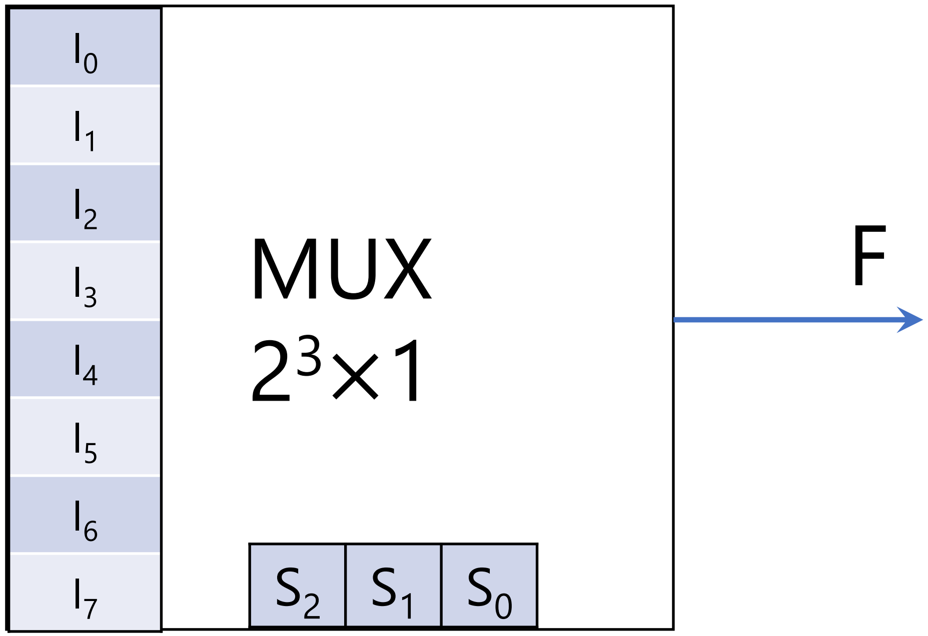
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

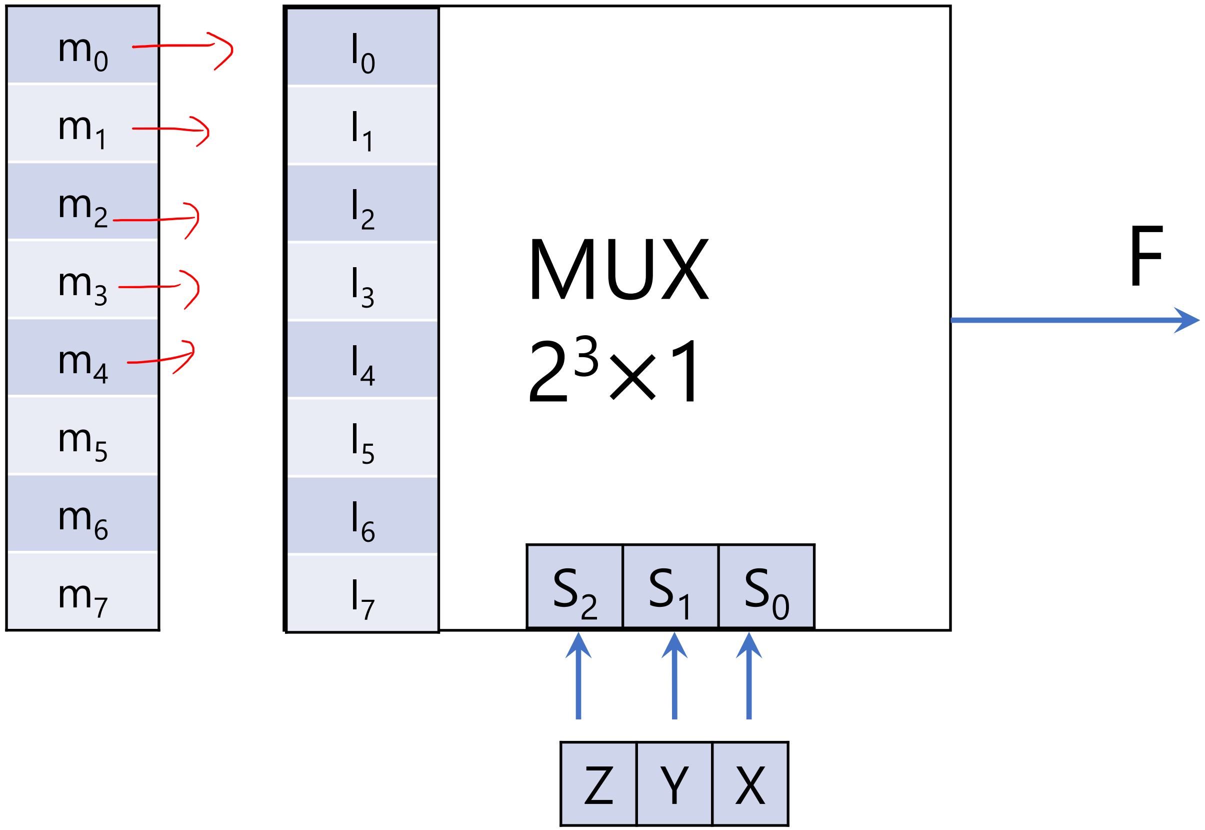
(b) Function table

FIGURE 4.25
Four-to-one-line multiplexer

Multiplexer
Boolean Function

$$F_{SoP} = \sum m(\ldots)$$
$$F_{PoS} = \prod M(\ldots)$$


I0
I1
I2
I3
I4
I5
I6
I7
MUX
2³×1
S2
S1
S0
F

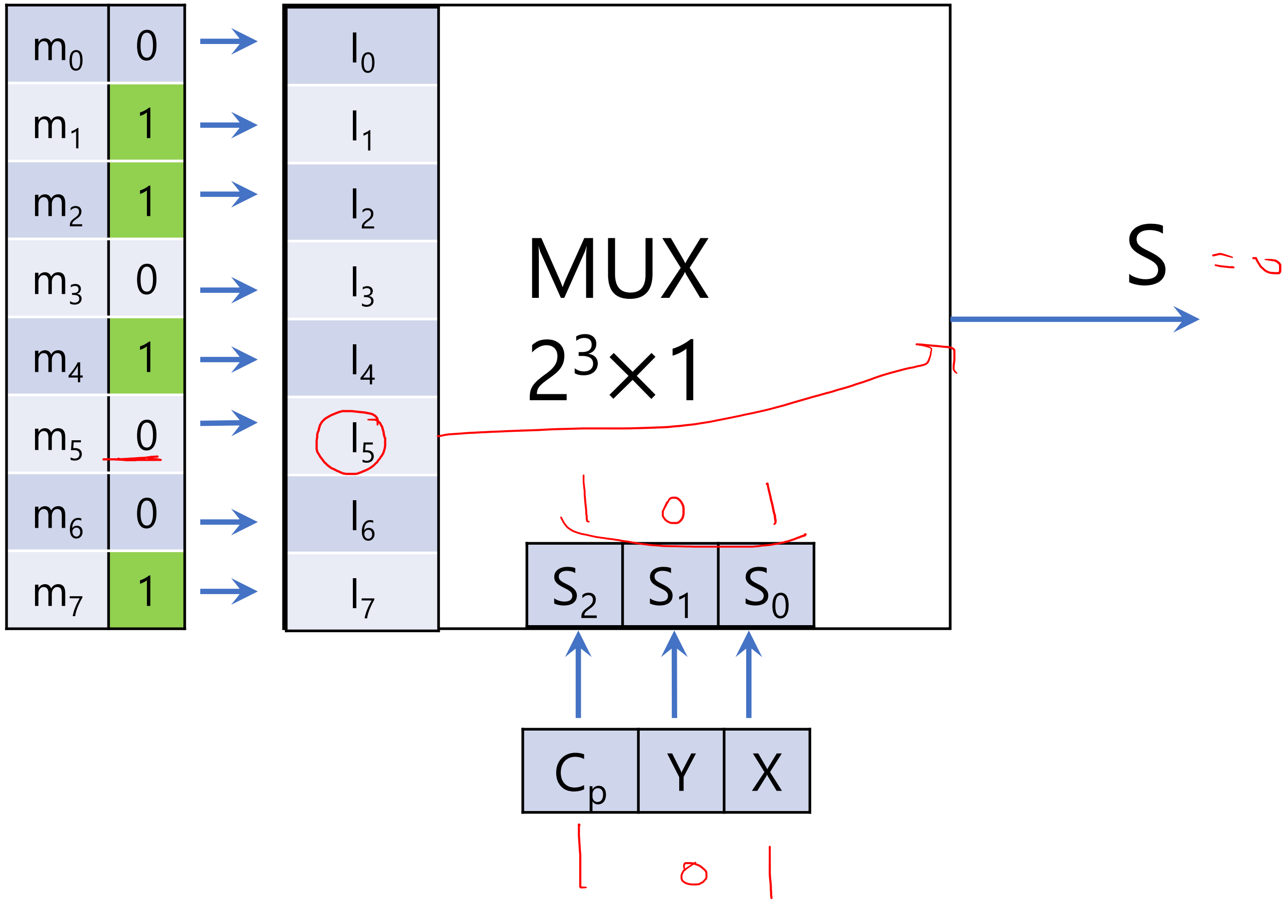


m_0
m_1
m_2
m_3
m_4
m_5
m_6
m_7
I_0
I_1
I_2
I_3
I_4
I_5
I_6
I_7
MUX
$2^3 \times 1$
F
S_2
S_1
S_0
Z
Y
X

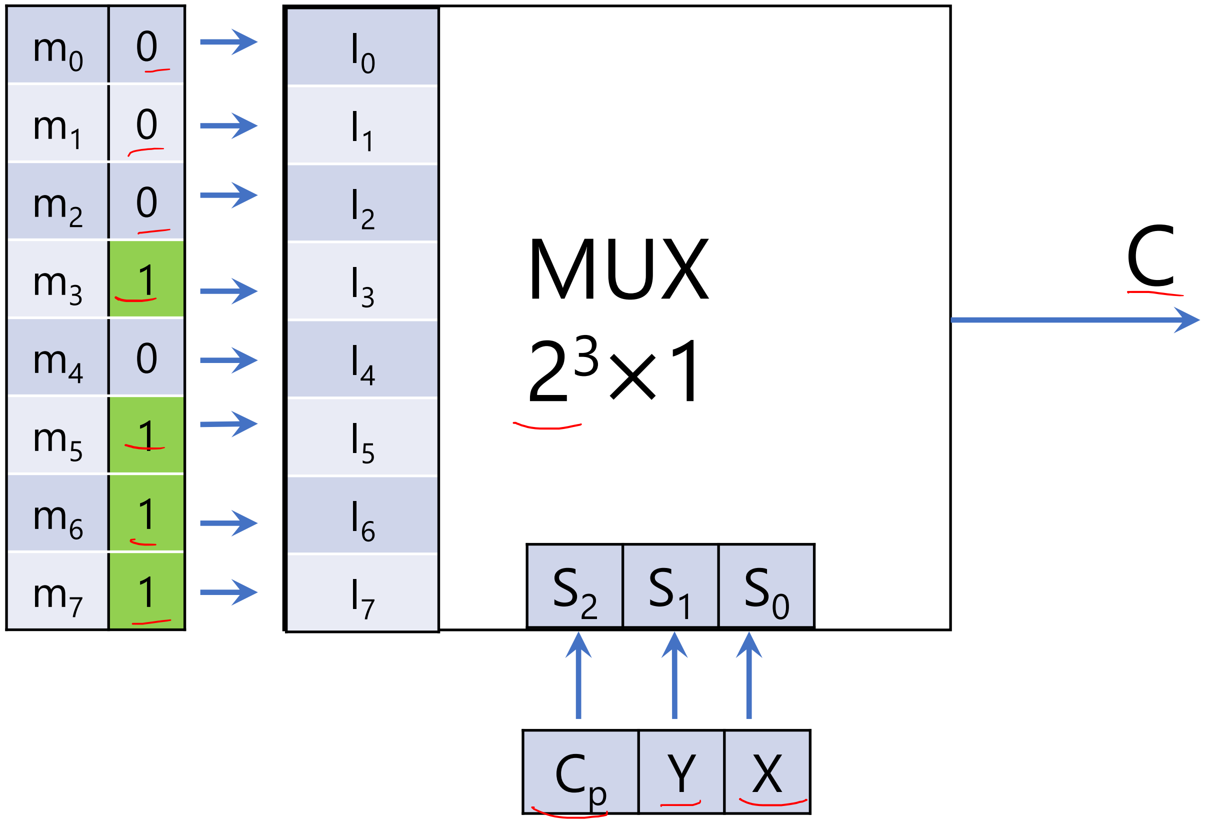
MUX
Full Adder

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

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


m0 0
m1 1
m2 1
m3 0
m4 1
m5 0
m6 0
m7 1
I0
I1
I2
I3
I4
I5
I6
I7
MUX
2^3×1
S
= 0
S2 S1 S0
1 0 1
Cp Y X
1 0 1

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

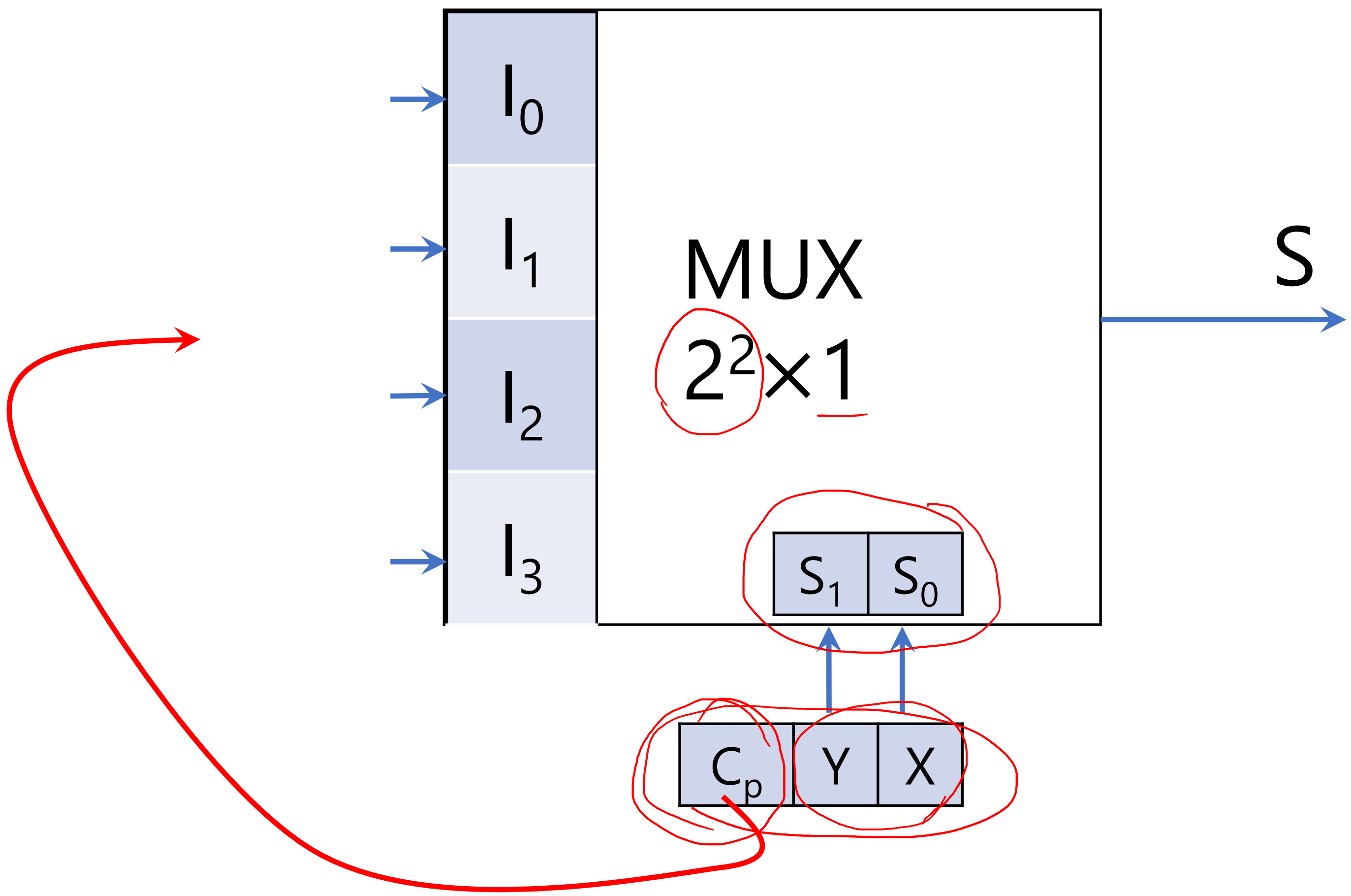


m0 0
m1 0
m2 0
m3 1
m4 0
m5 1
m6 1
m7 1
I0
I1
I2
I3
I4
I5
I6
I7
MUX
2^3×1
C
S2 S1 S0
Cp Y X

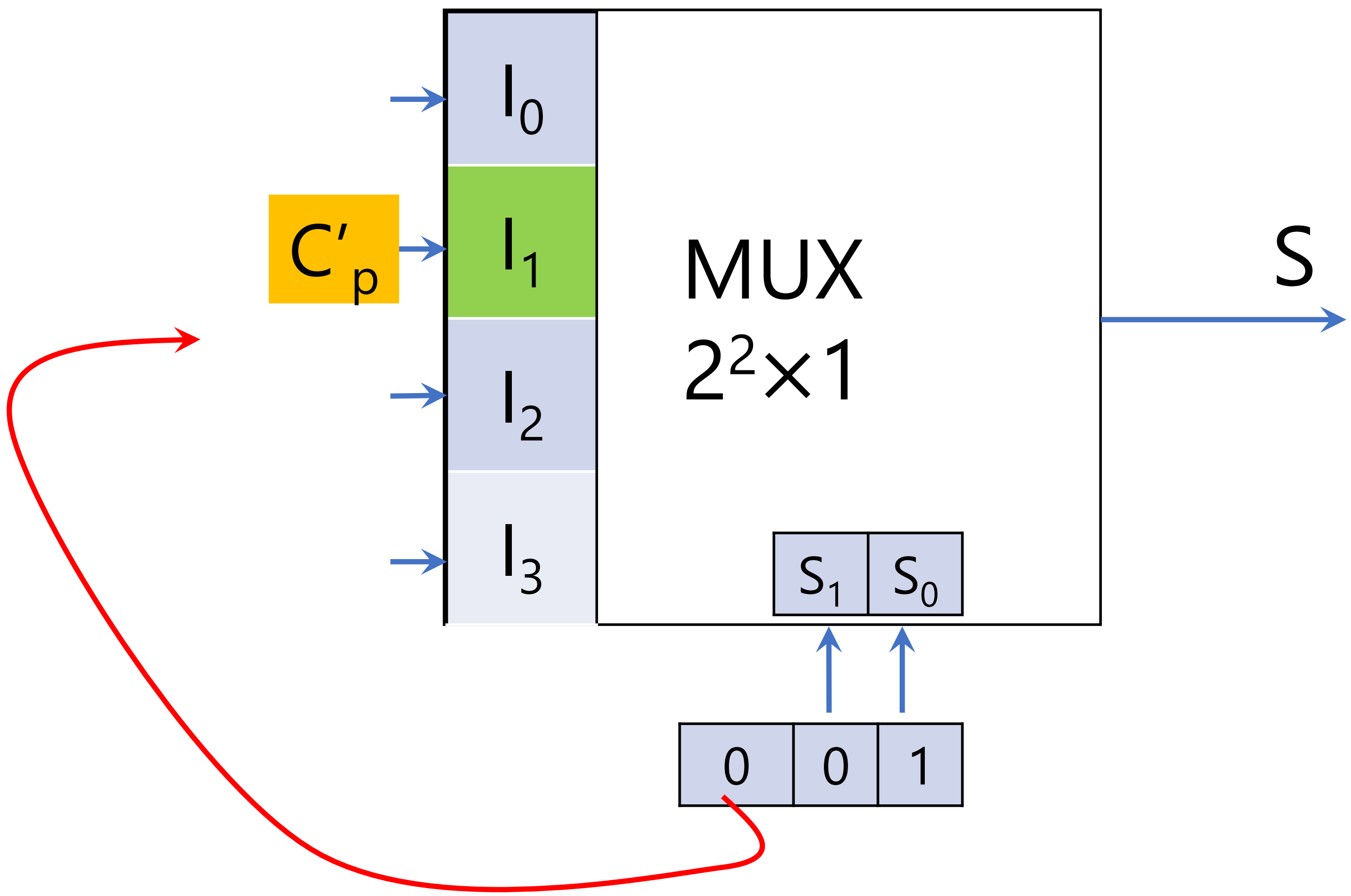
$$C = \sum m(3,5,6,7)$$

Multiplexer
Boolean Function II

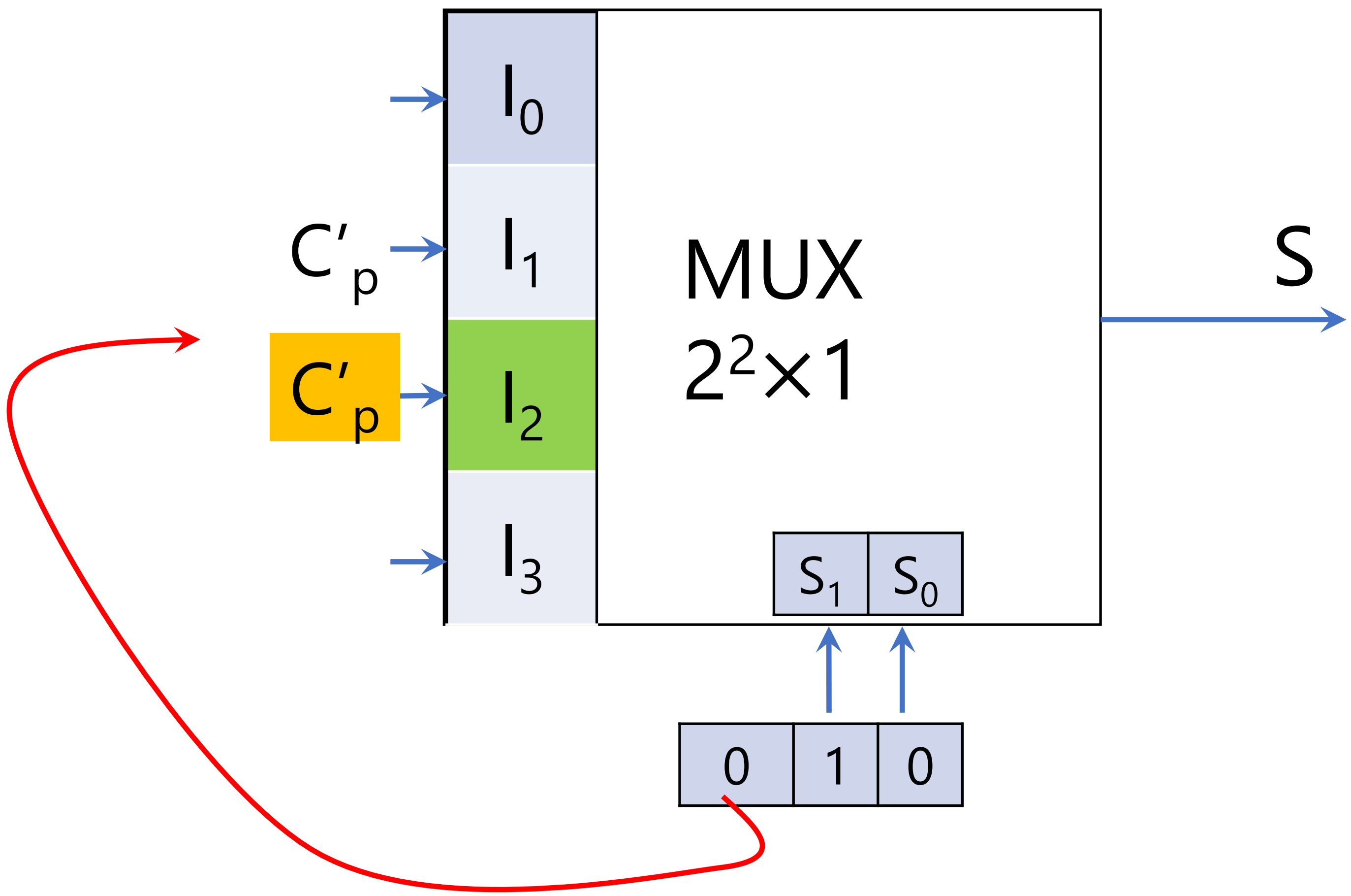
Book: Page 161



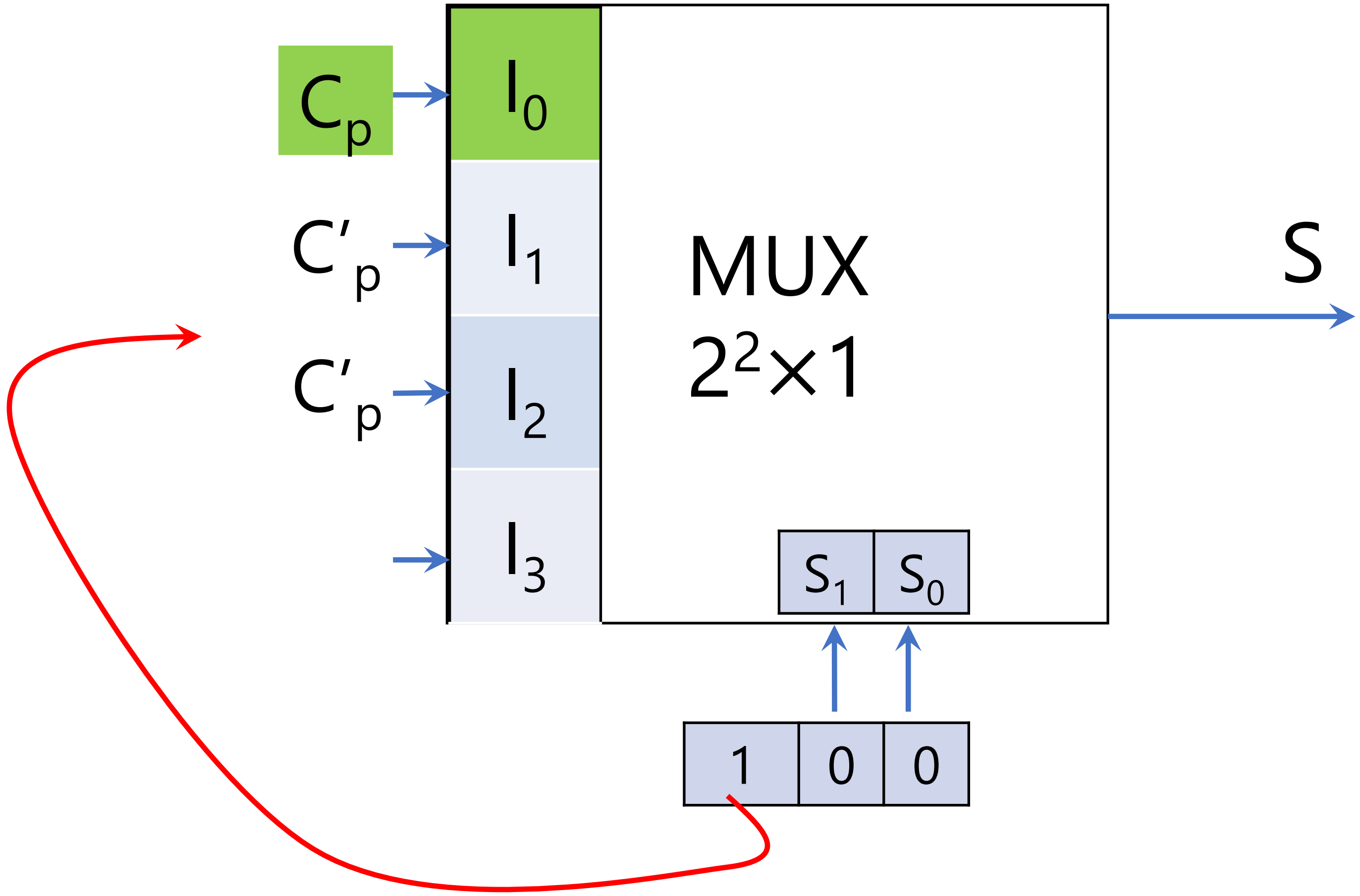
$$S = \sum m(1,2,4,7)$$



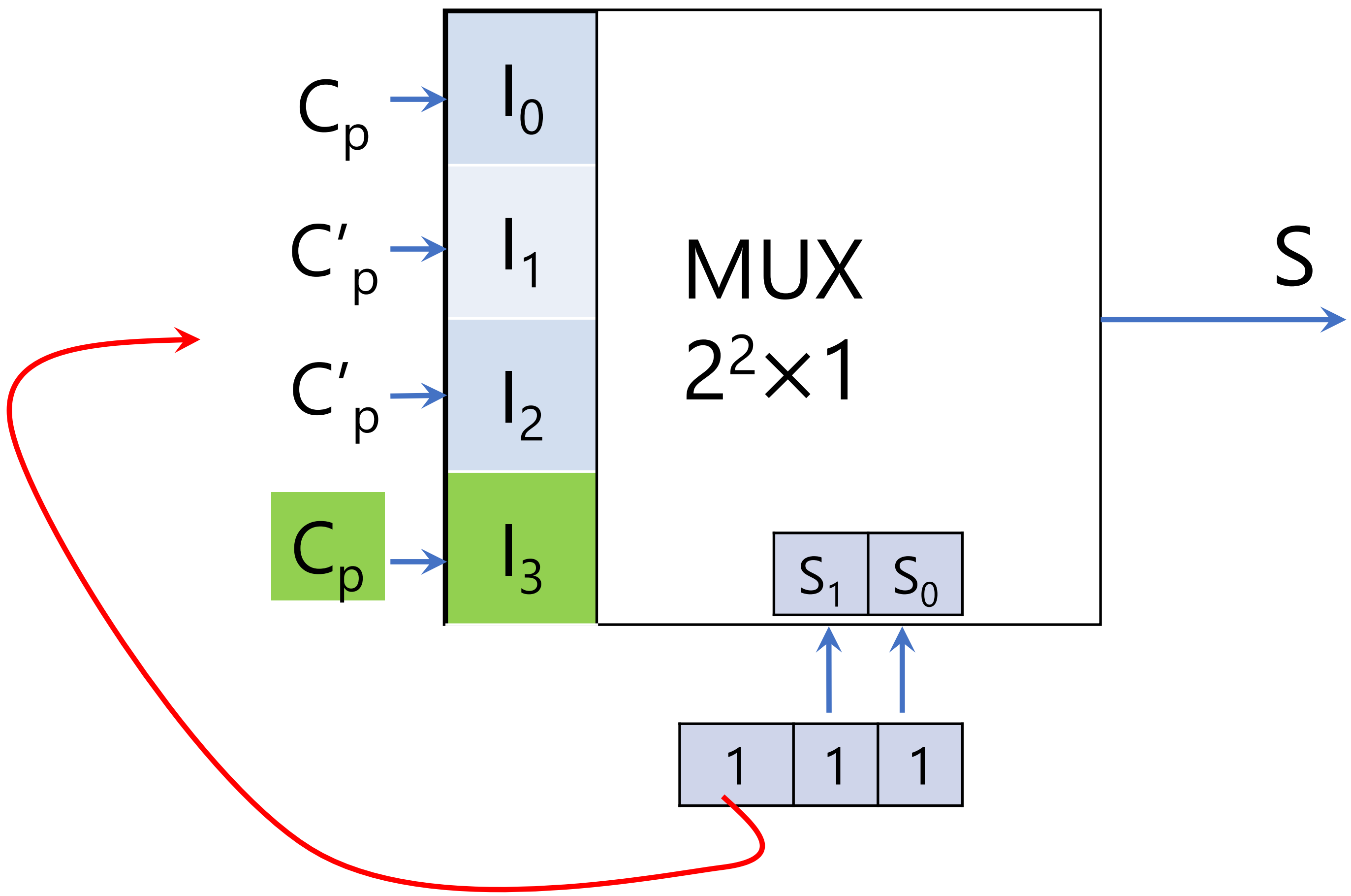
$$S = \sum m(1,2,4,7)$$



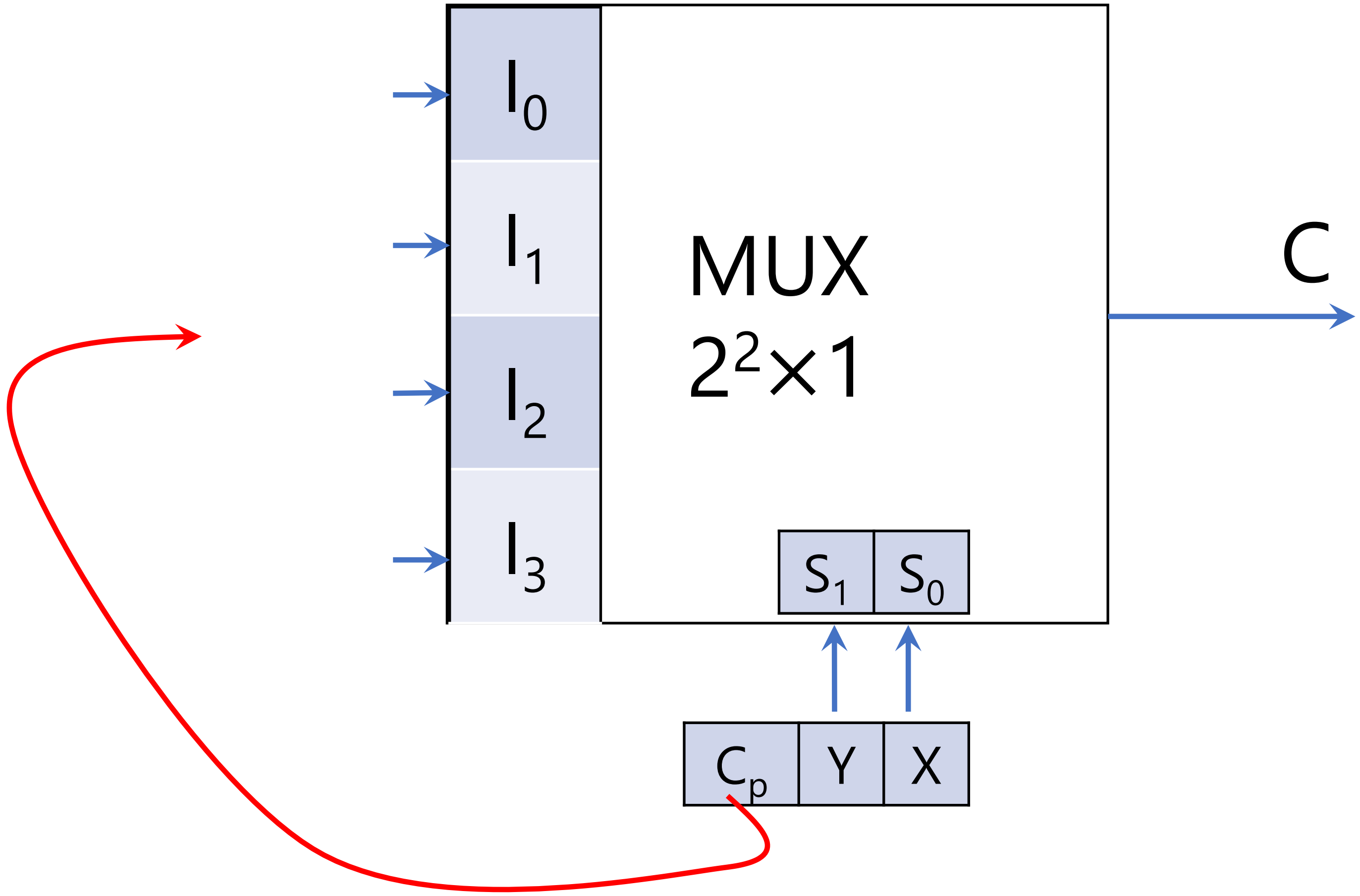
$$S = \sum m(1,2,4,7)$$



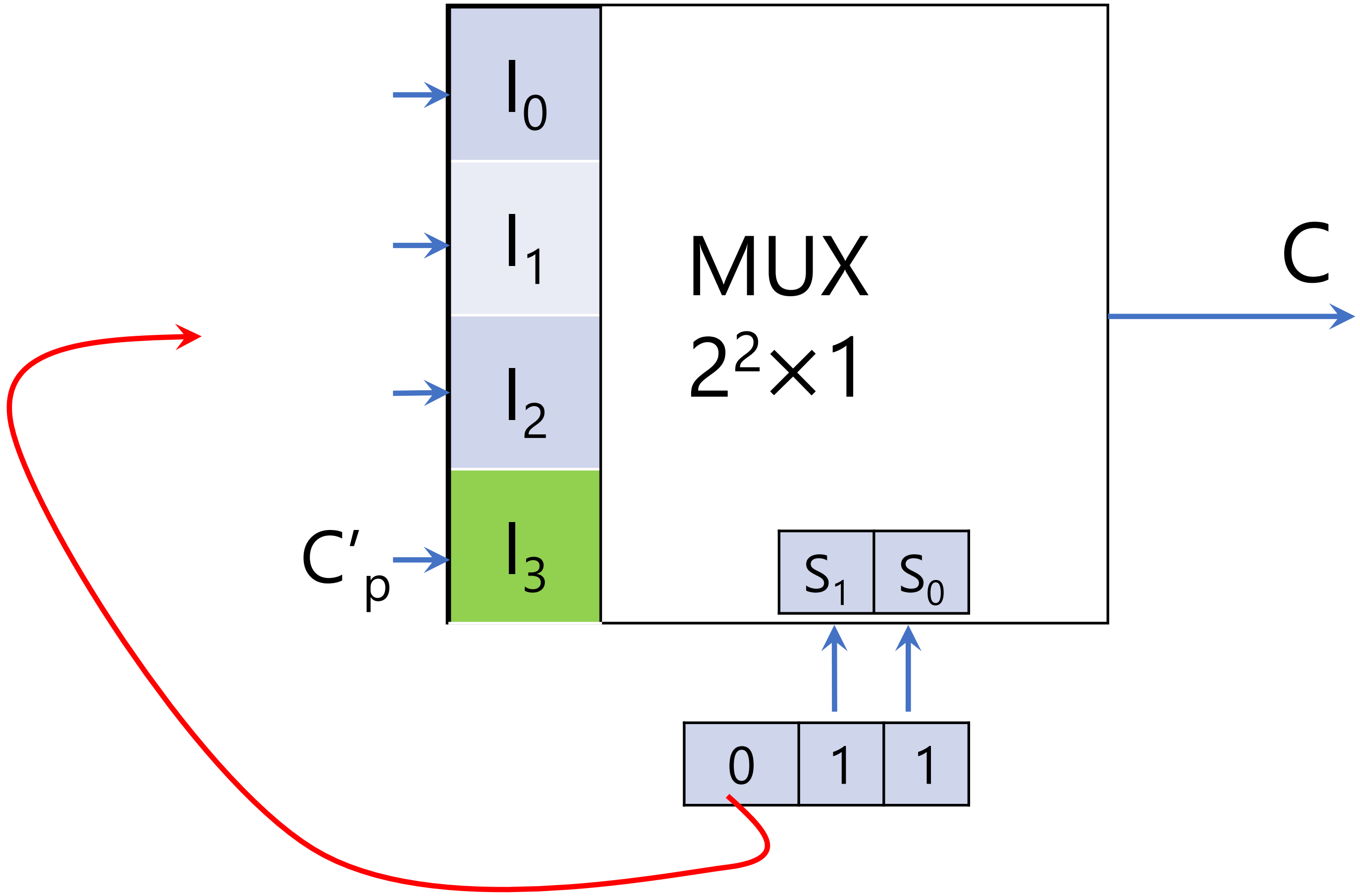
$$S = \sum m(1,2,4,7)$$



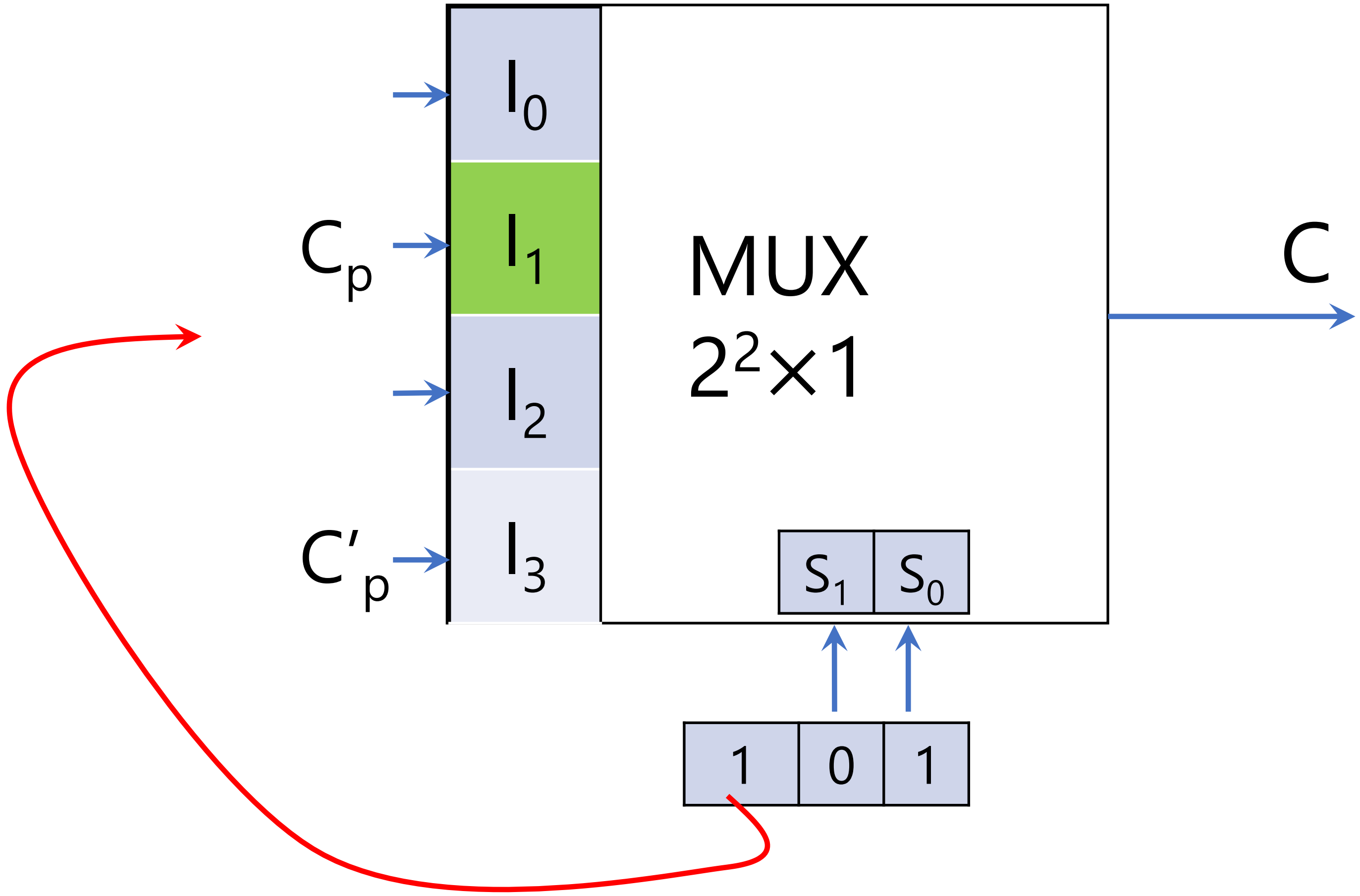
$$S = \sum m(1,2,4,7)$$



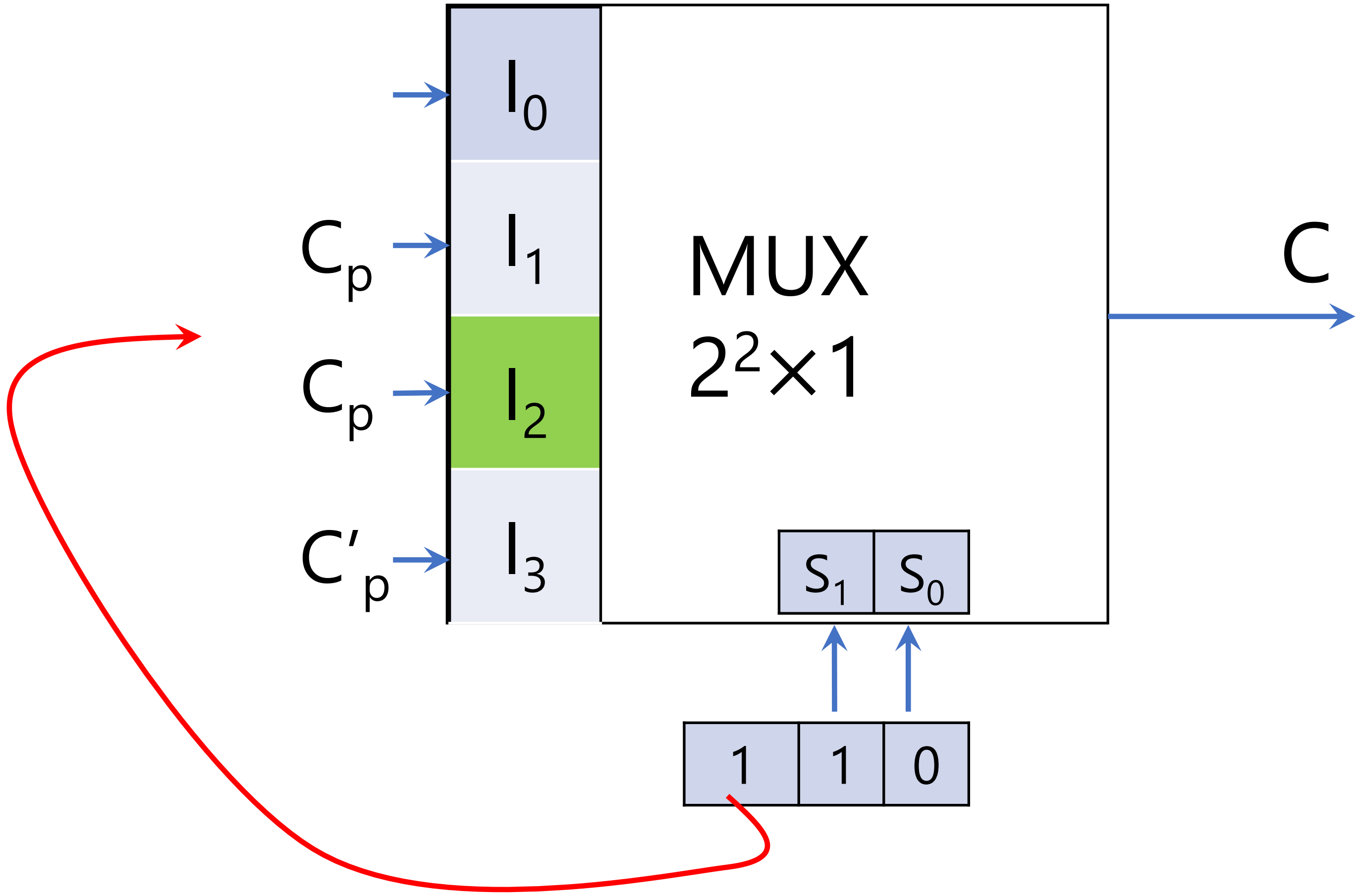
$$C = \sum m(3,5,6,7)$$



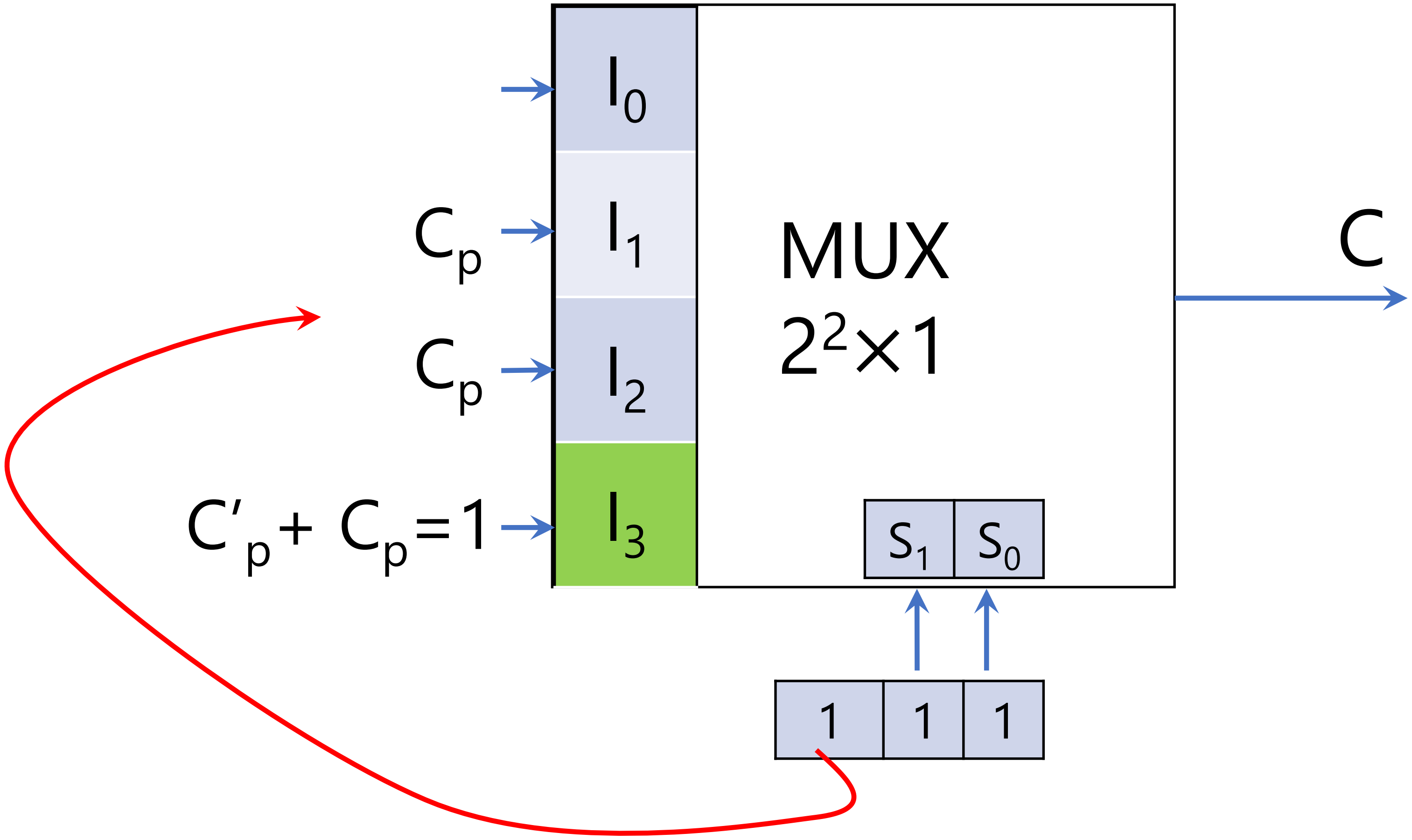
$$C = \sum m(3,5,6,7)$$



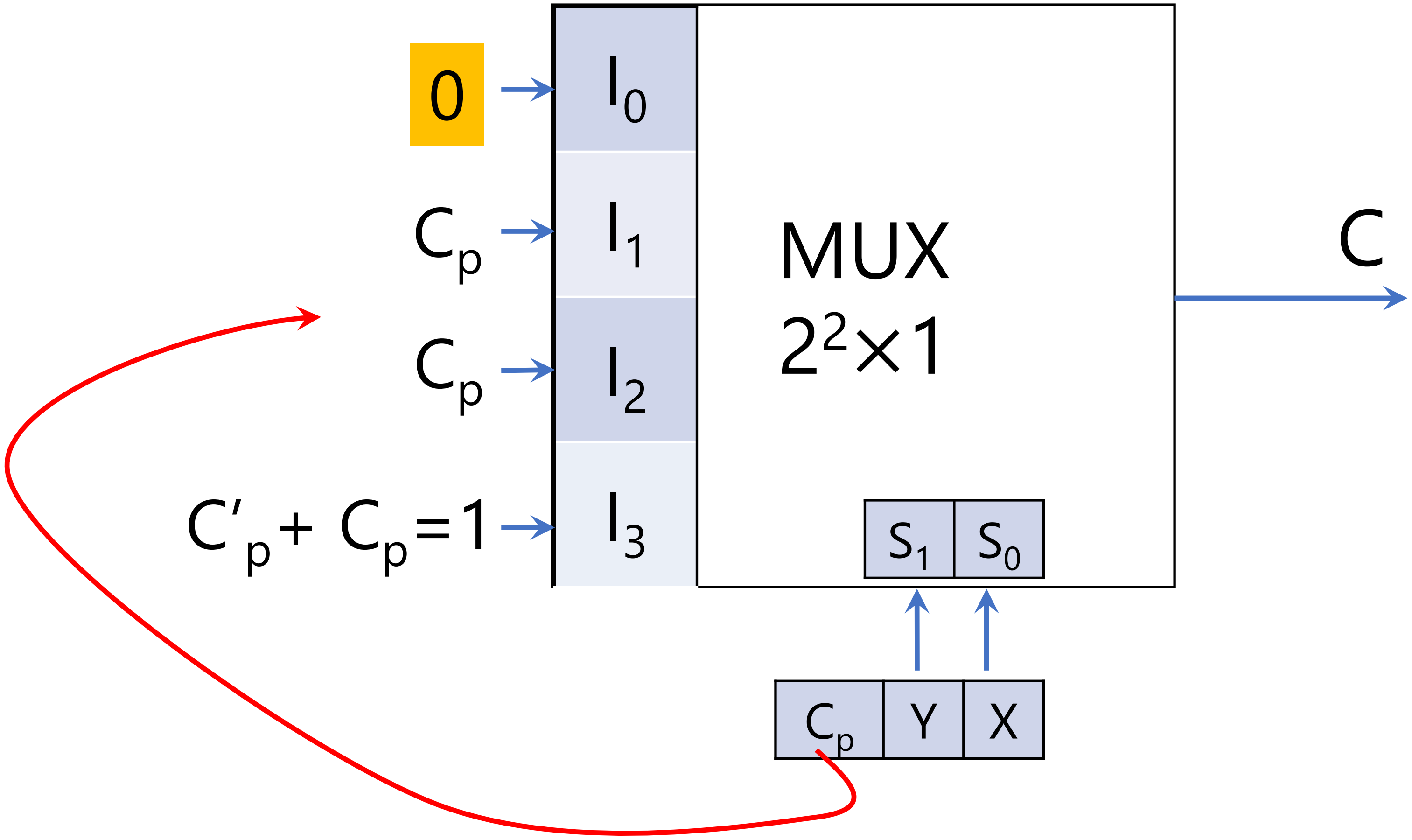
$$C = \sum m(3,5,6,7)$$



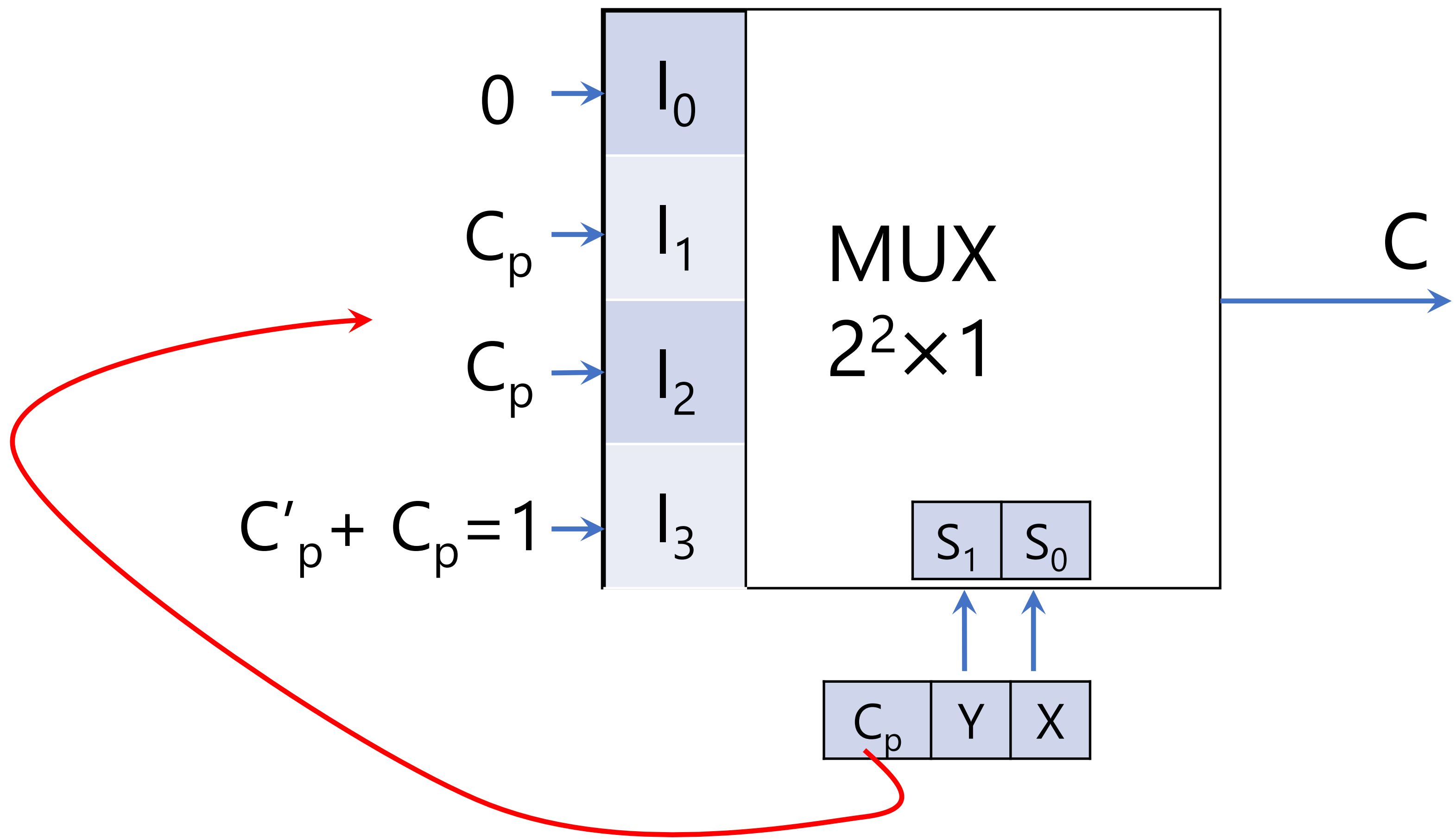
$$C = \sum m(3,5,6,7)$$



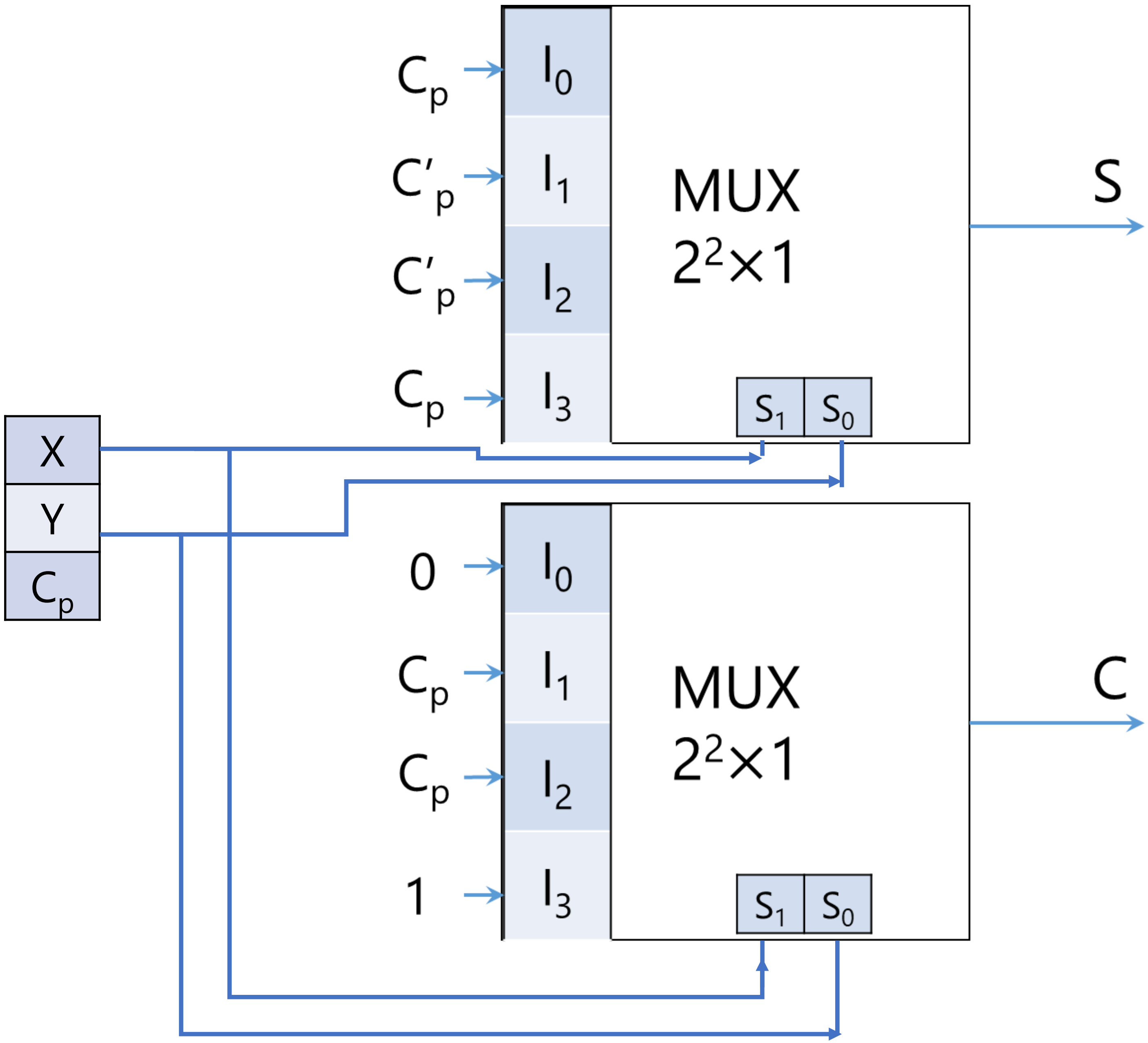
$$C = \sum m(3,5,6,7)$$



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



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



X
Y
Cp
Cp
C'p
C'p
Cp
I0
I1
I2
I3
MUX
2²×1
S1
S0
S
0
Cp
Cp
1
I0
I1
I2
I3
MUX
2²×1
S1
S0
C

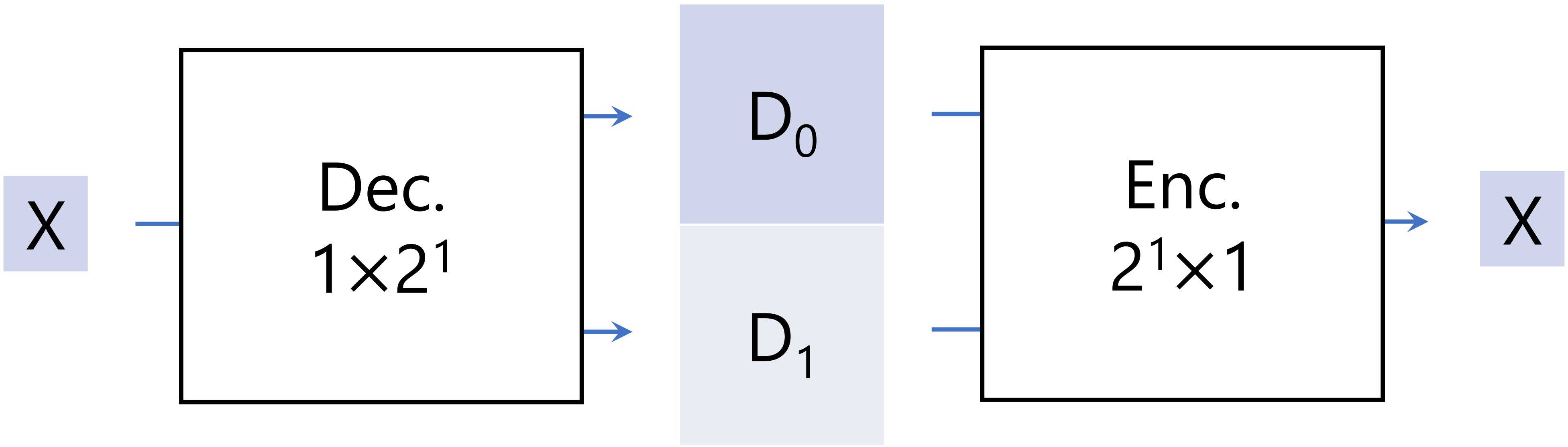
Multiplexer
Three-State Gates + Decoders

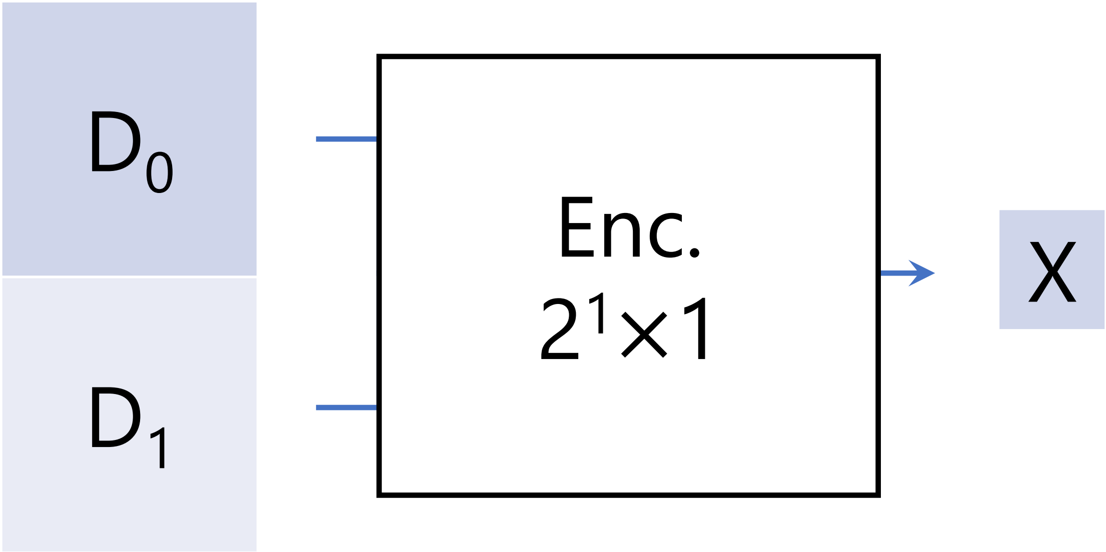
Book: Page 162-164


Data Transmission
Decoder, **Encoder**
Multiplexer (MUX, MPX), De-Multiplexer (Demux)

Encoder

Encoder
1-hot to Binary


X
Dec.
1×2^1
D0
D1
Enc.
2^1×1
X



D_0
D_1
Enc.
$2^1 \times 1$
X

D_1	D_0	F_1
0	0	X
0	1	0
1	0	1
1	1	X

X: Don't Care Conditions

D_1	D_0	F_1
0	0	X
0	1	0
1	0	1
1	1	X

D_1 \ D_0	0	1
0	X m_0	0 m_1
1	1 m_2	X m_3

D_1	D_0	F_1
0	0	X
0	1	0
1	0	1
1	1	X

D_1 \ D_0	0	1
0	X (m_0)	0 (m_1)
1	1 (m_2)	X (m_3)

$$F_1 = D'_0$$

D_1	D_0	F_1
0	0	X
0	1	0
1	0	1
1	1	X

D_1 \ D_0	0	1
0	X m_0	0 m_1
1	1 m_2	X m_3

$$F_1 = D_1$$

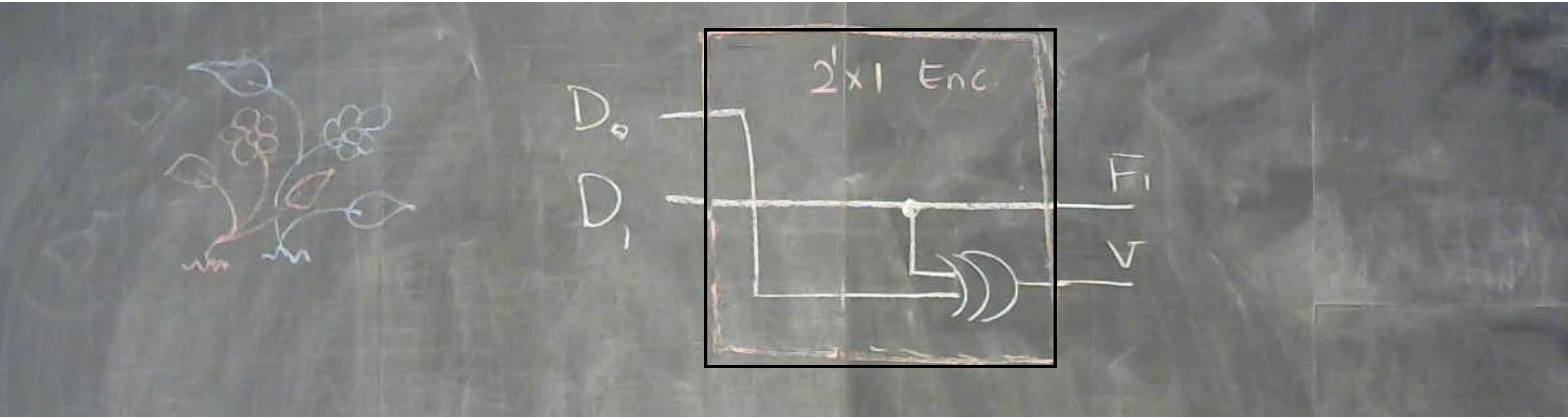
D_1	D_0	F_1	V
0	0	X	0
0	1	0	1
1	0	1	1
1	1	X	0

D_1 \ D_0	0	1
0	X (m_0)	0 (m_1)
1	1 (m_2)	X (m_3)

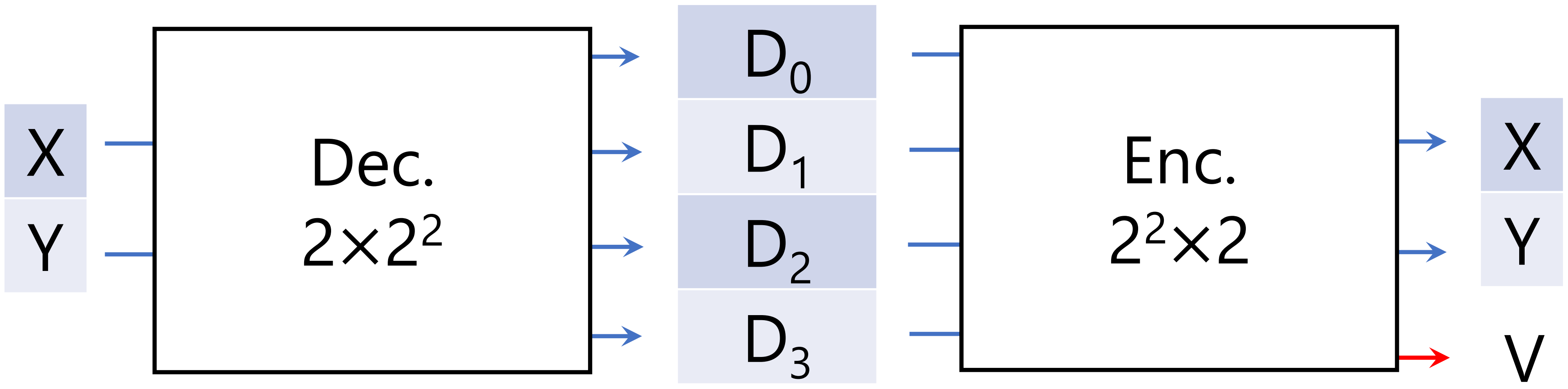
$$F_1 = D_1$$

D_1 \ D_0	0	1
0	0 (m_0)	1 (m_1)
1	1 (m_2)	0 (m_3)

$$V = D_0D'_1 + D'_0D_1$$
$$= D_0 \oplus D_1$$



2¹x1 Enc
D0
D1
F1
V


X
Y
Dec.
2×2²
D_0
D_1
D_2
D_3
Enc.
2²×2
X
Y
V

D_3	D_2	D_1	D_0	$F_2=Y$	$F_1=X$	$F_3=V$
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	x	x	0
0	1	0	0	1	0	1
0	1	0	1	x	x	0
0	1	1	0	x	x	0
0	1	1	1	x	x	0
1	0	0	0	1	1	1
1	0	0	1	x	x	0
1	0	1	0	x	x	0
1	0	1	1	x	x	0
1	1	0	0	x	x	0
1	1	0	1	x	x	0
1	1	1	0	x	x	0
1	1	1	1	x	x	0

D_3D_2 \ D_1D_0	00	01	11	10
00	X m_0	0 m_1	X m_3	1 m_2
01	0 m_4	X m_5	X m_7	X m_6
11	X m_{12}	X m_{13}	X m_{15}	X m_{14}
10	1 m_8	X m_9	X m_{11}	X m_{10}

$$F_1 = X = D_1 + D_3$$

D_3D_2 \ D_1D_0	00	01	11	10
00	X m_0	0 m_1	X m_3	0 m_2
01	1 m_4	X m_5	X m_7	X m_6
11	X m_{12}	X m_{13}	X m_{15}	X m_{14}
10	1 m_8	X m_9	X m_{11}	X m_{10}

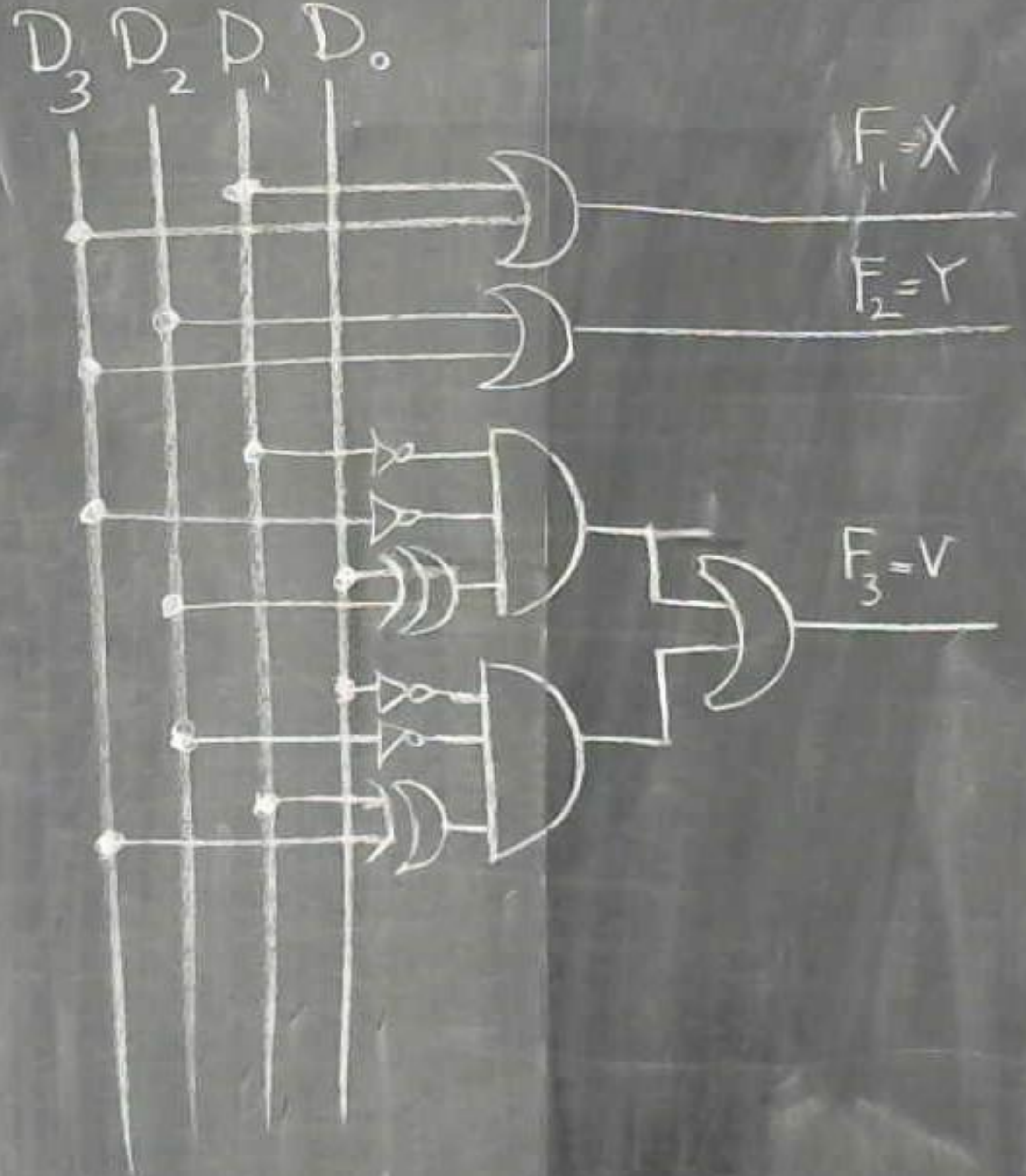
$$F_2 = Y = D_2 + D_3$$

D_3D_2 \ D_1D_0	00	01	11	10
00	0 m_0	1 m_1	0 m_3	1 m_2
01	1 m_4	0 m_5	0 m_7	0 m_6
11	0 m_{12}	0 m_{13}	0 m_{15}	0 m_{14}
10	1 m_8	0 m_9	0 m_{11}	0 m_{10}

$$F_3 = V = D'_3D'_2D'_1D_0 + D'_3D_2D'_1D'_0 + D'_3D'_2D_1D'_0 + D_3D'_2D'_1D'_0$$

$$= D'_3D'_1(D'_2D_0 + D_2D'_0) + D'_2D'_0(D'_3D_1 + D_3D'_1)$$

$$= D'_3D'_1(D_2 \oplus D_0) + D'_2D'_0(D_3 \oplus D_1)$$



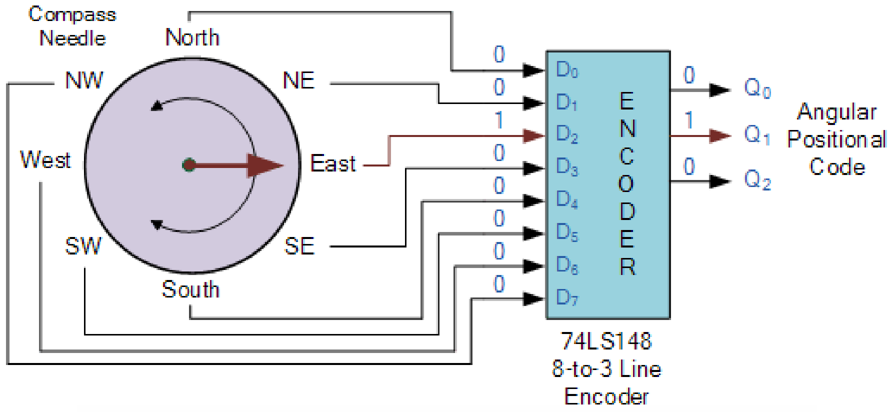
D3 D2 D1 D0
F1=X
F2=Y
F3=V

Priority Encoder

at home!

Positional Encoders

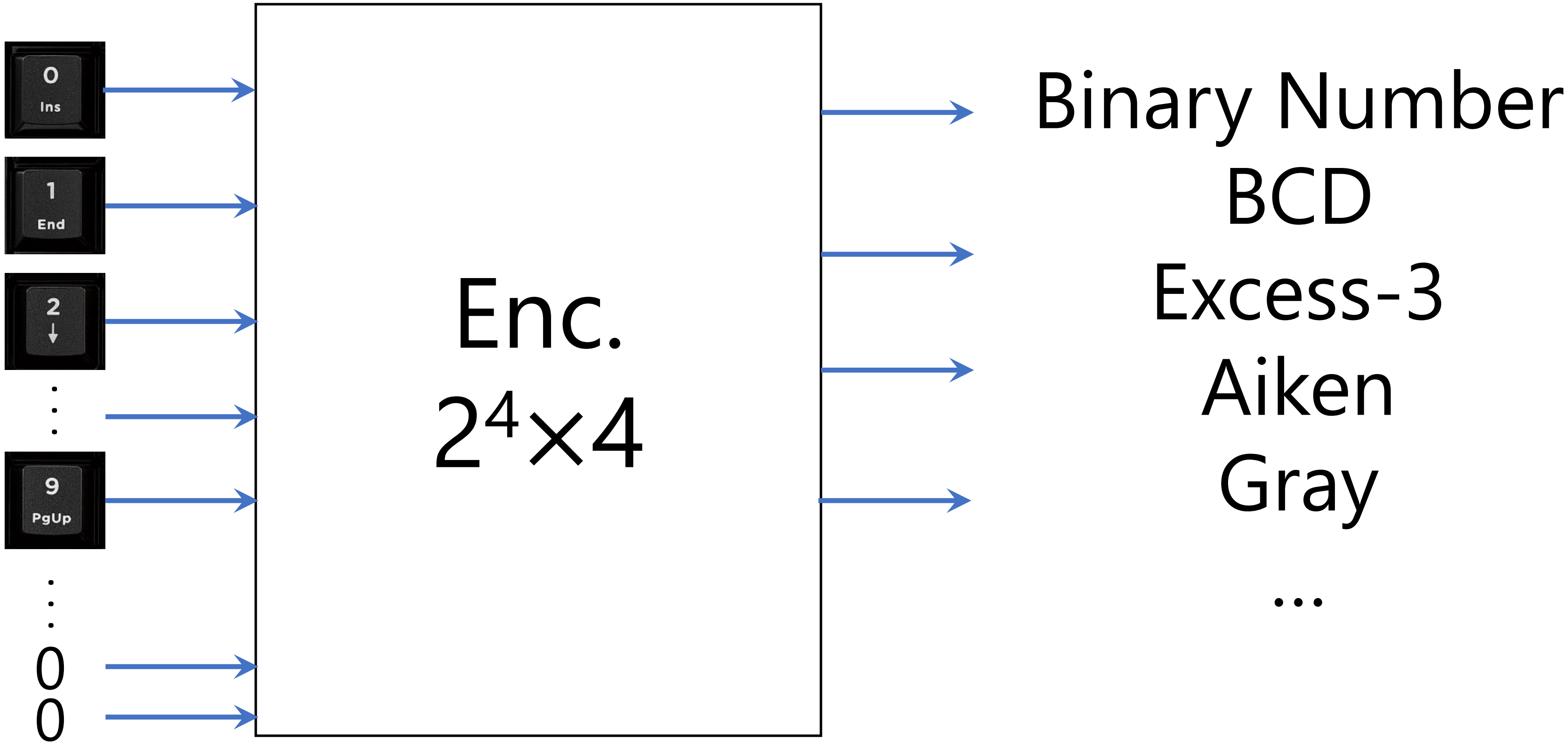
Priority Encoder Navigation

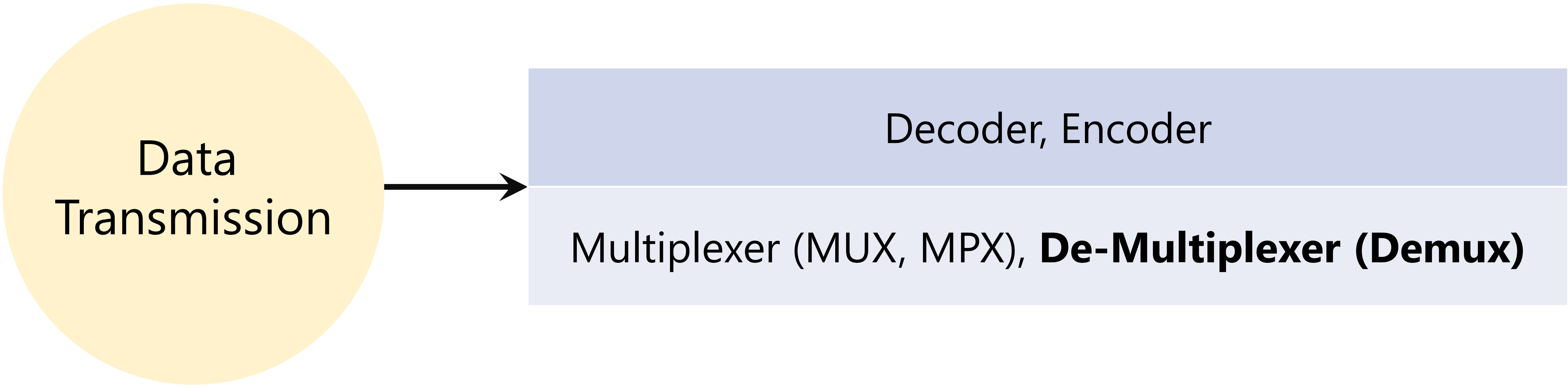


Compass Direction	Binary Output		
	Q_0	Q_1	Q_2
North	0	0	0
North-East	0	0	1
East	0	1	0
South-East	0	1	1
South	1	0	0
South-West	1	0	1
West	1	1	0
North-West	1	1	1

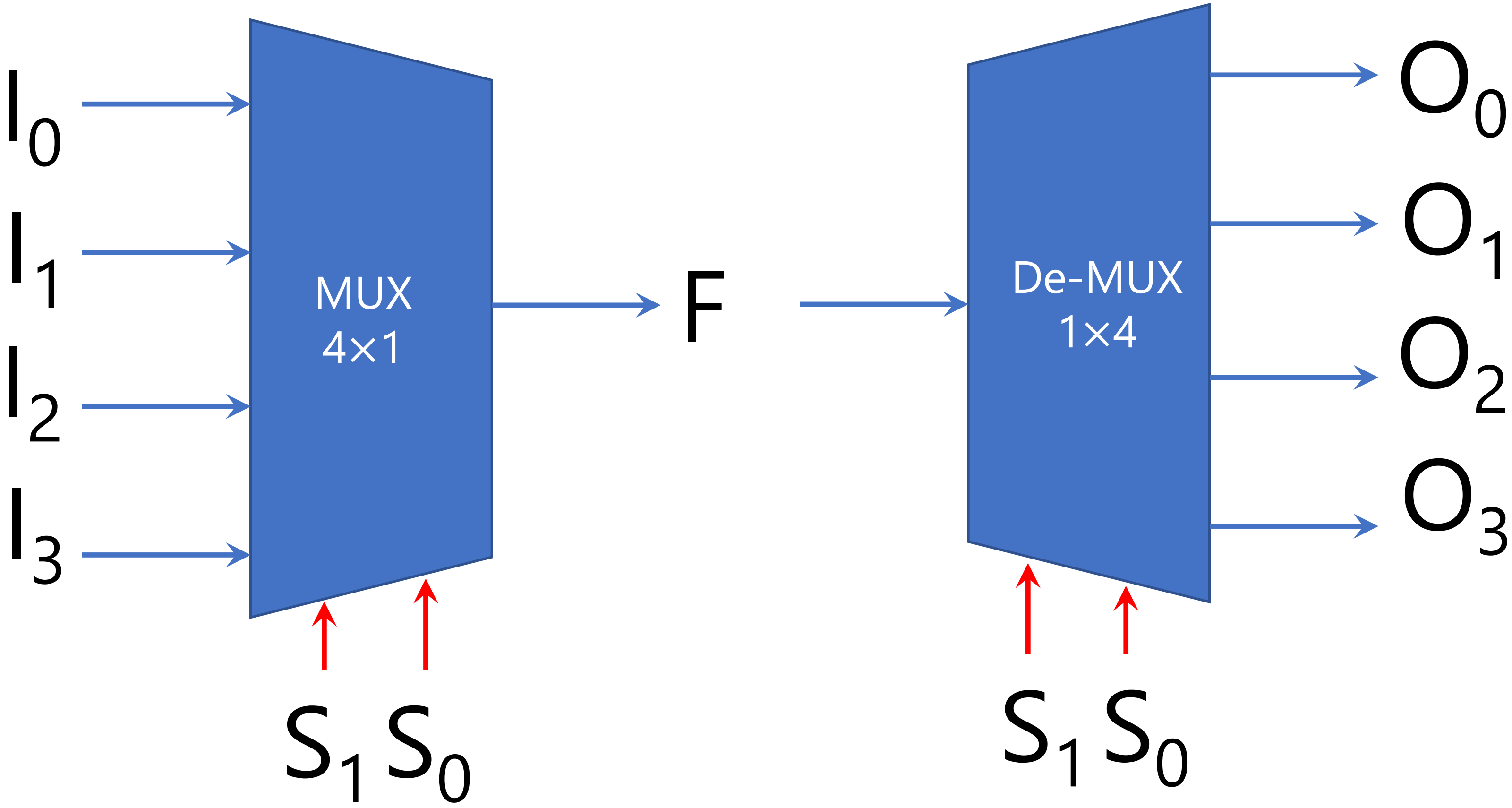
https://www.electronics-tutorials.ws/

Keyboard Encoders

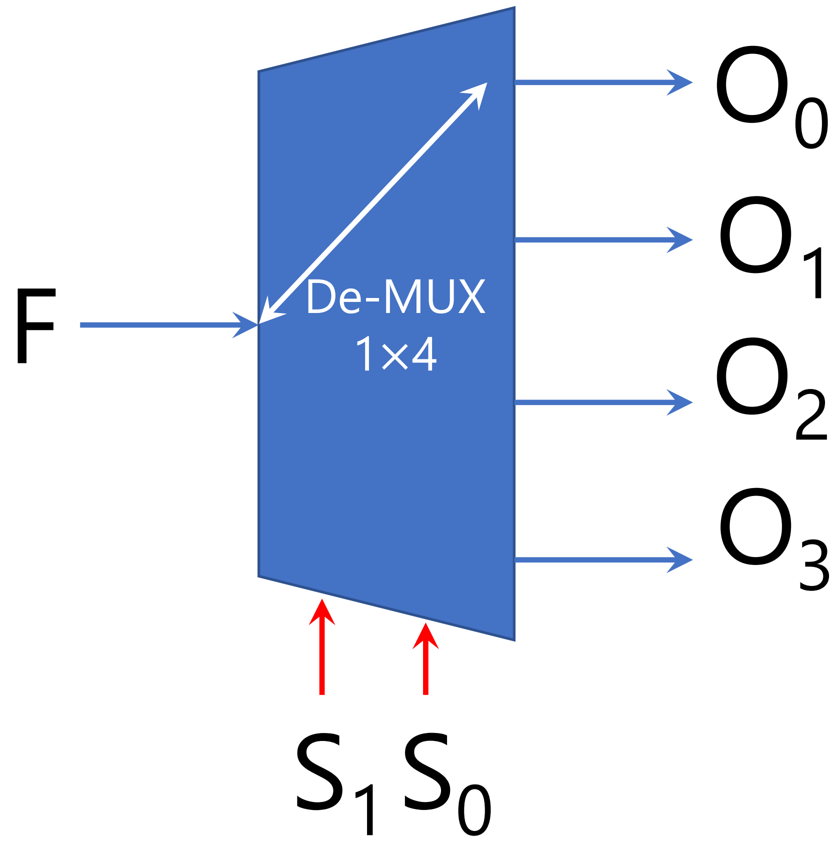

0
Ins
1
End
2
9
PgUp
0
0
Enc.
$2^4 \times 4$
Binary Number
BCD
Excess-3
Aiken
Gray
...


Data Transmission
Decoder, Encoder
Multiplexer (MUX, MPX), **De-Multiplexer (Demux)**

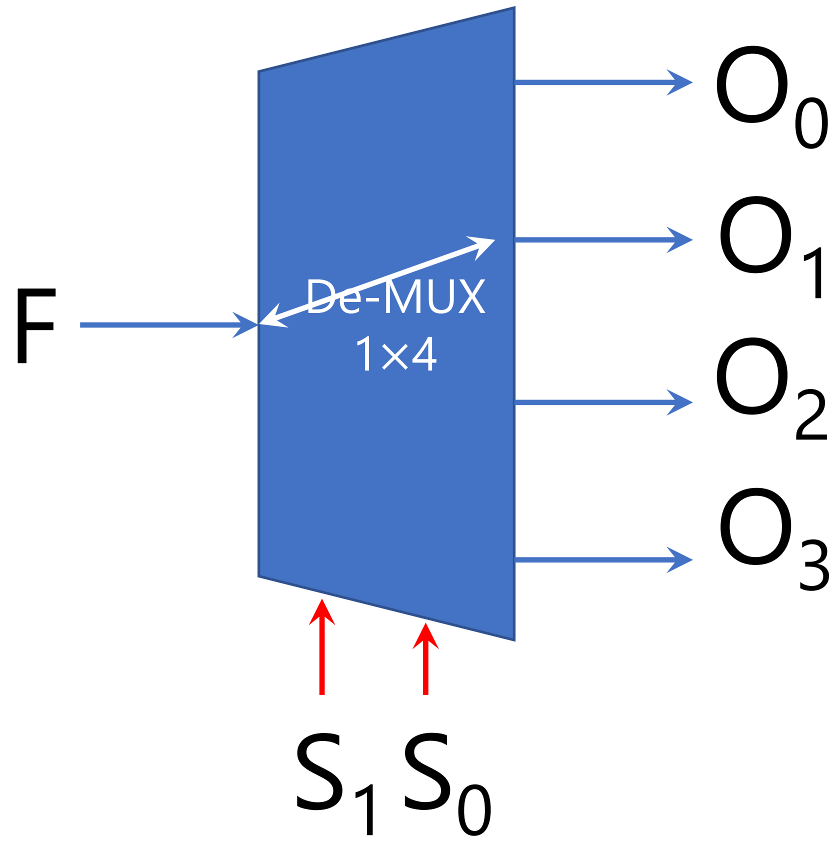
De-multiplexer



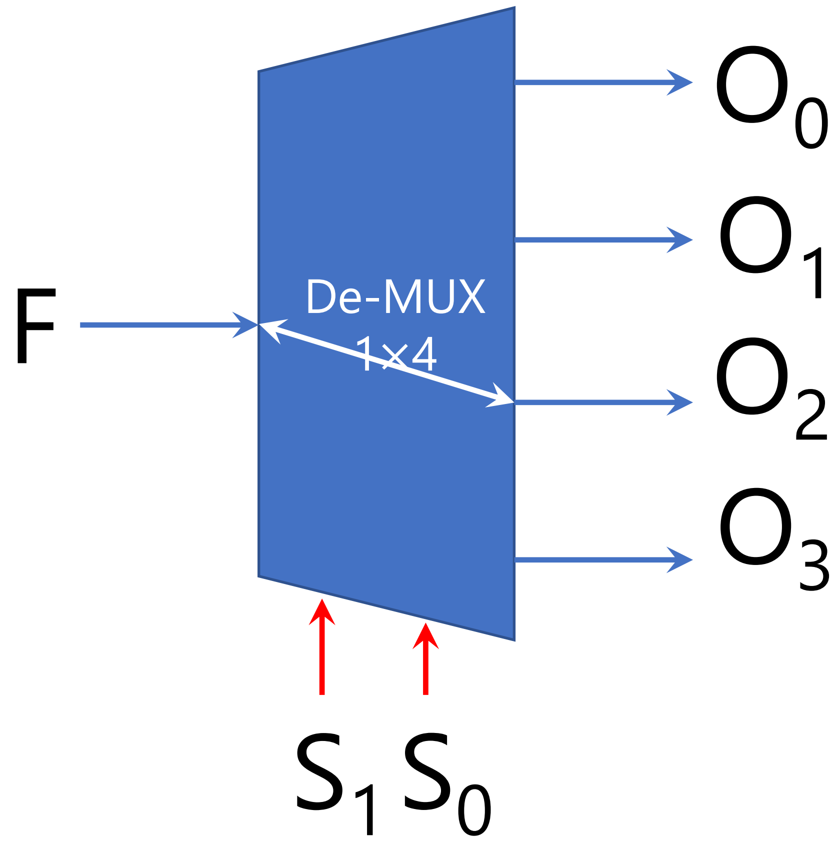
I0
I1
I2
I3
MUX
4×1
F
De-MUX
1×4
O0
O1
O2
O3
S1 S0
S1 S0



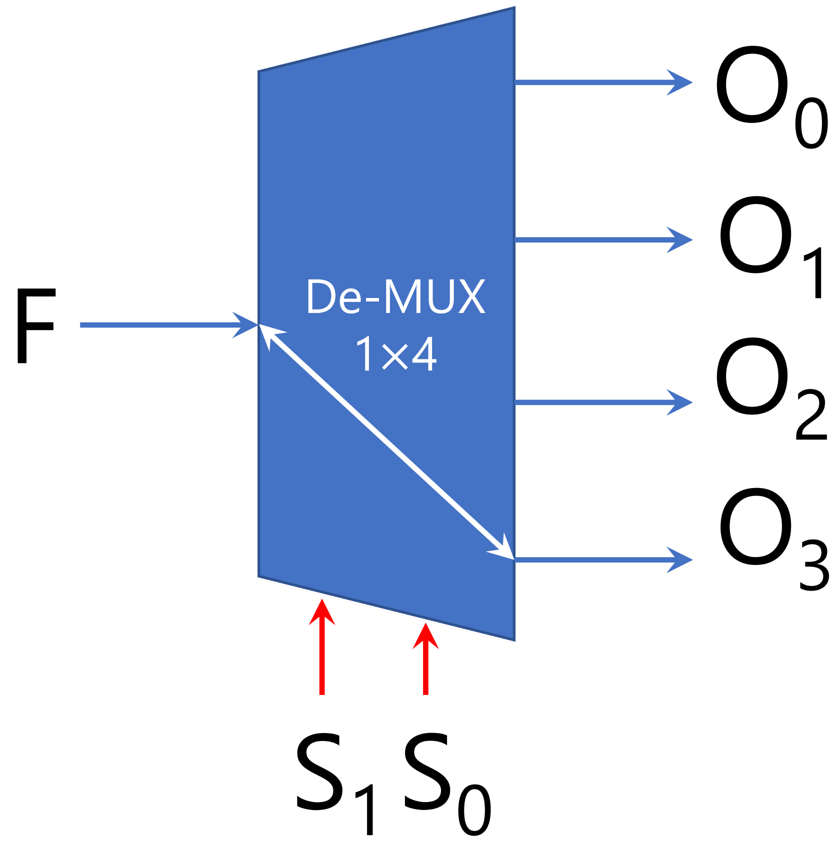
S_1	S_0	F	O_0	O_1	O_2	O_3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1
1	1	1	0	0	0	0



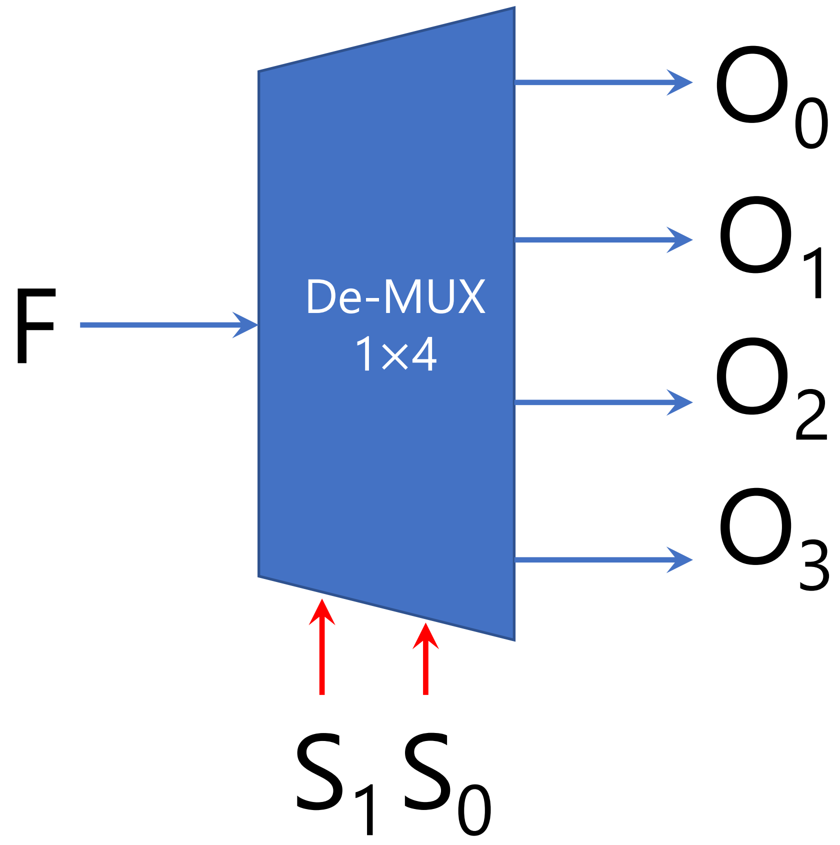
S_1	S_0	F	O_0	O_1	O_2	O_3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1
1	1	1	0	0	0	0



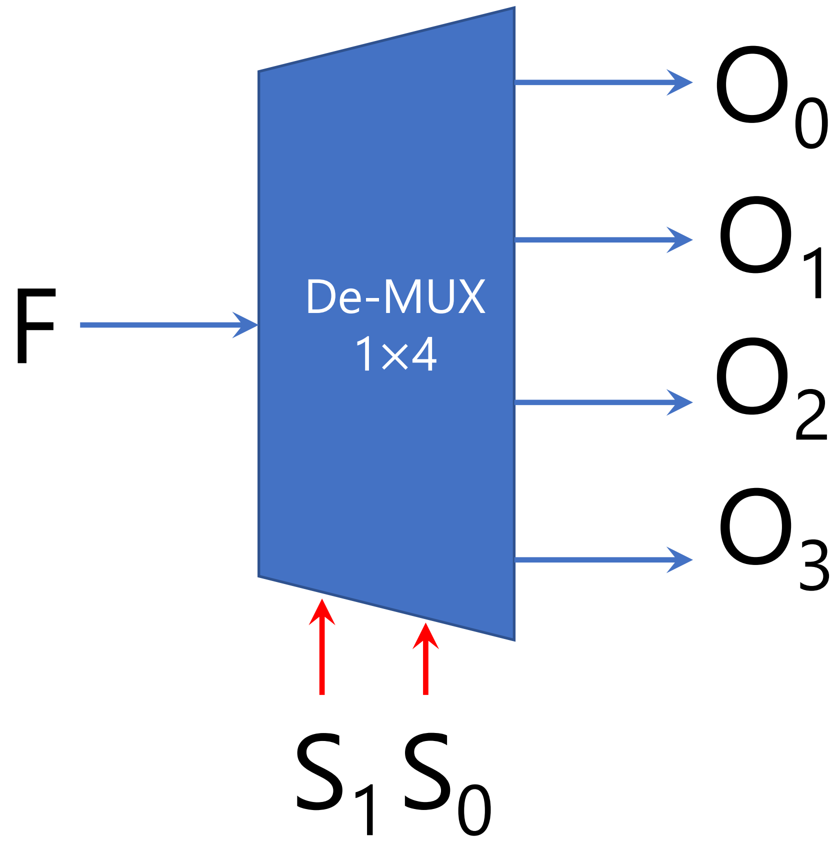
S_1	S_0	F	O_0	O_1	O_2	O_3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1
1	1	1	0	0	0	0



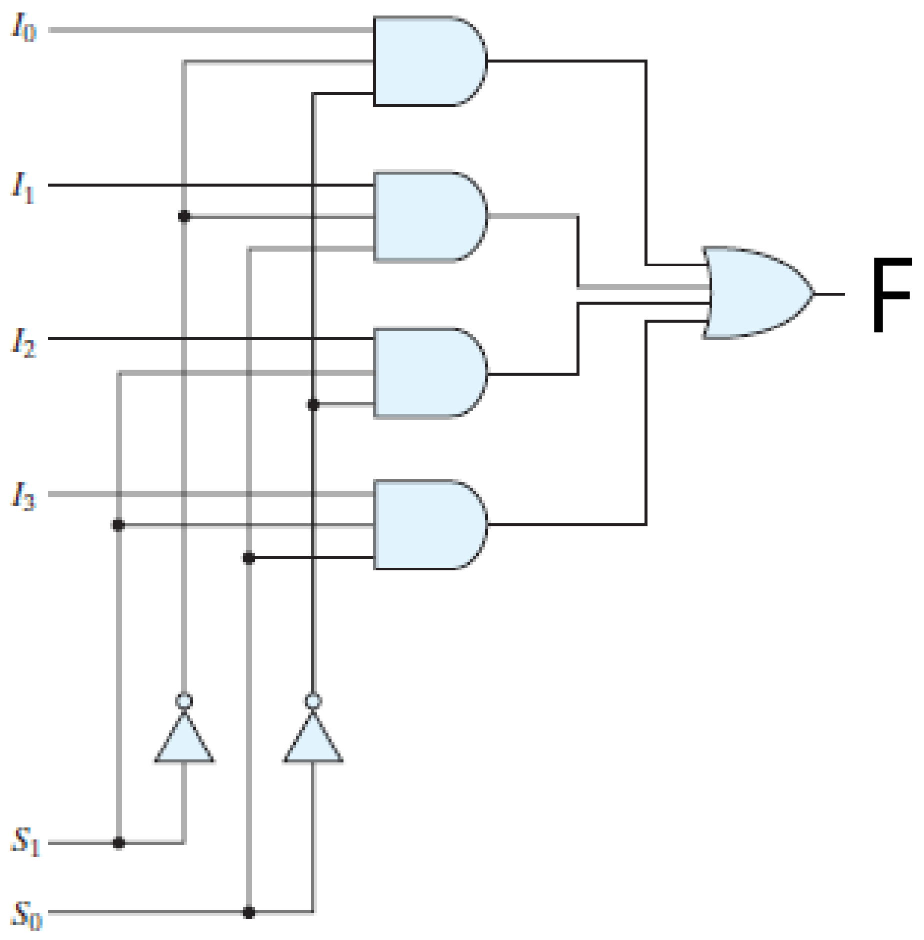
S_1	S_0	F	O_0	O_1	O_2	O_3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1



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

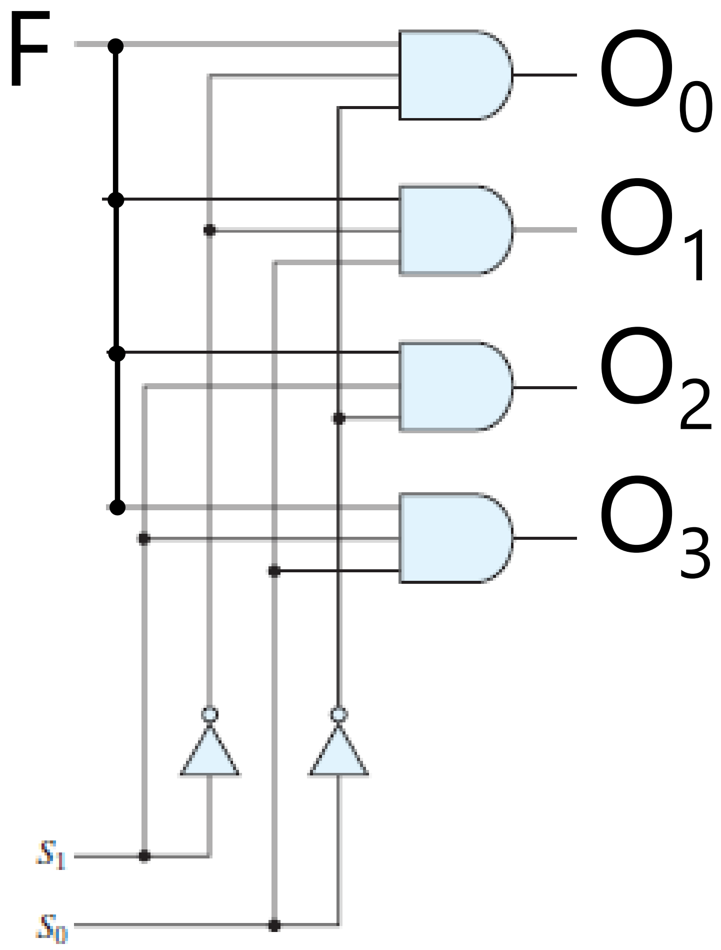


S_1	S_0	$O_0 = S'_1S'_0F$	$O_1 = S'_1S_0F$	$O_2 = S_1S'_0F$	$O_3 = S_1S_0F$
0	0	F	0	0	0
0	1	0	F	0	0
1	0	0	0	F	0
1	1	0	0	0	F



(a) Logic diagram

FIGURE 4.25
Four-to-one-line multiplexer



1-to-4 De-mux

De-multiplexer = Decoder w/ Enable input

How come?!