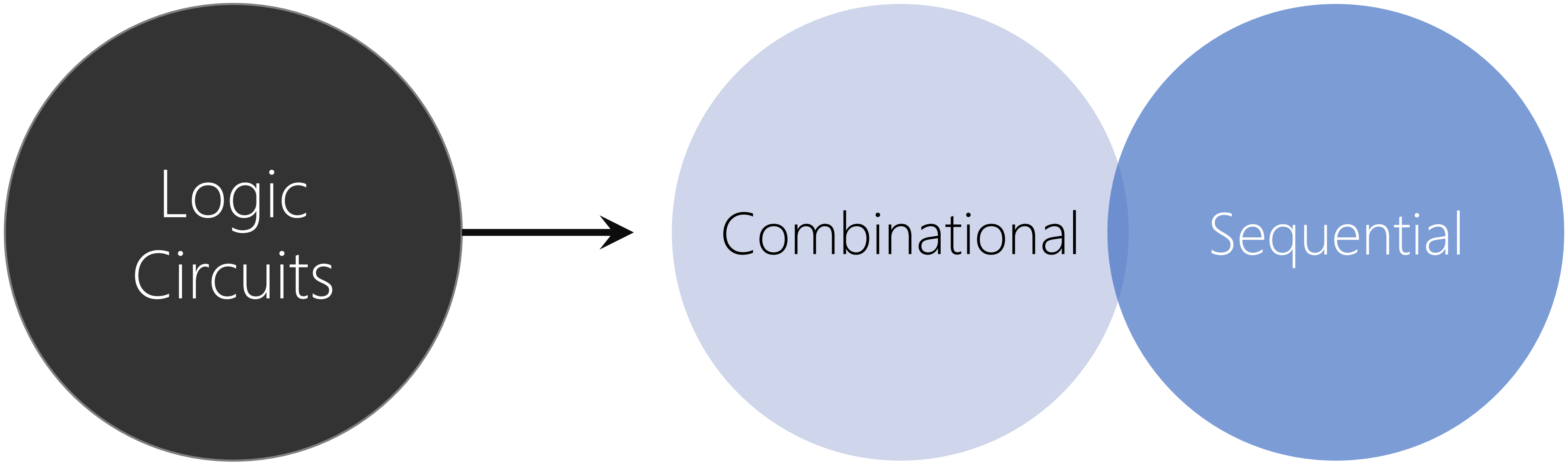

Logic Circuits
Combinational
Sequential

Chapter 4 Combinational Logic

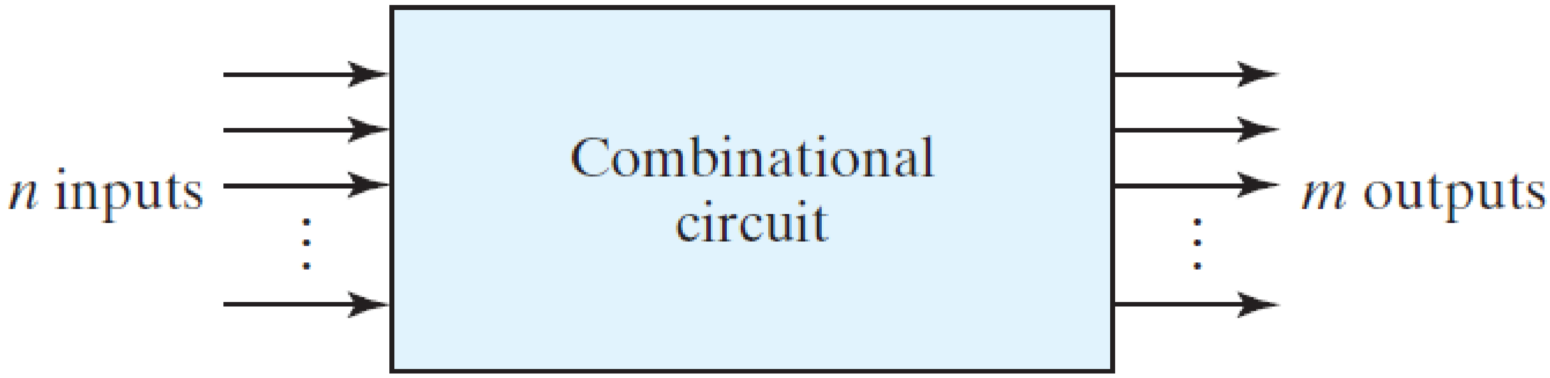
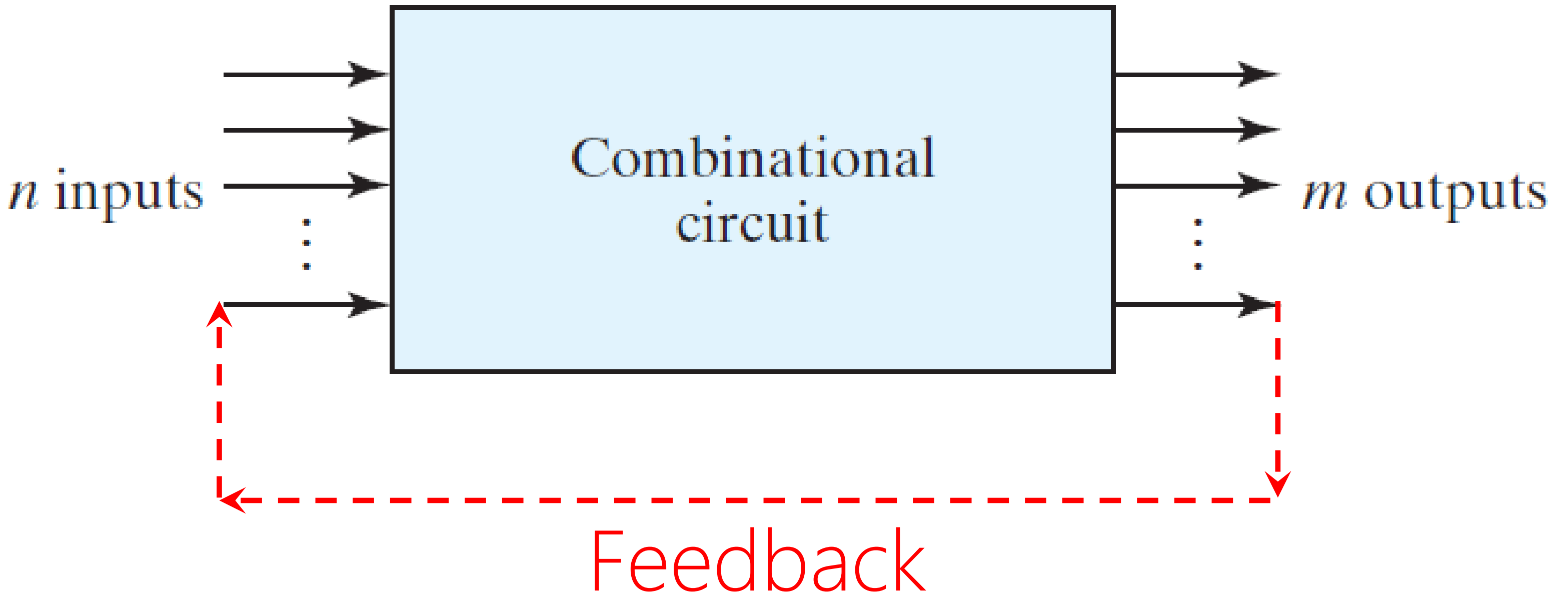
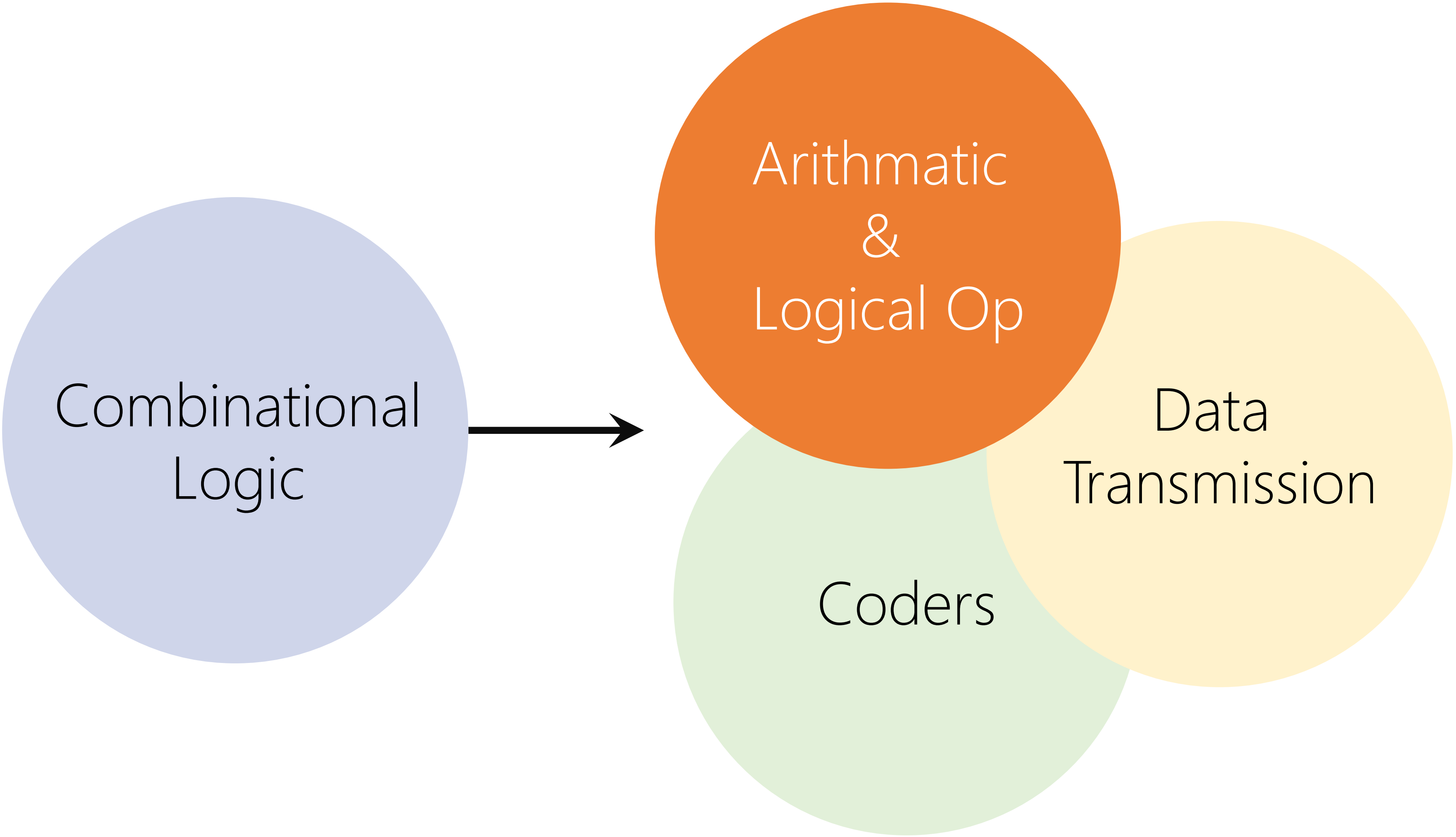


FIGURE 4.1
Block diagram of combinational circuit

Sequential Logic




Combinational Logic
Arithmatic & Logical Op
Data Transmission
Coders

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?

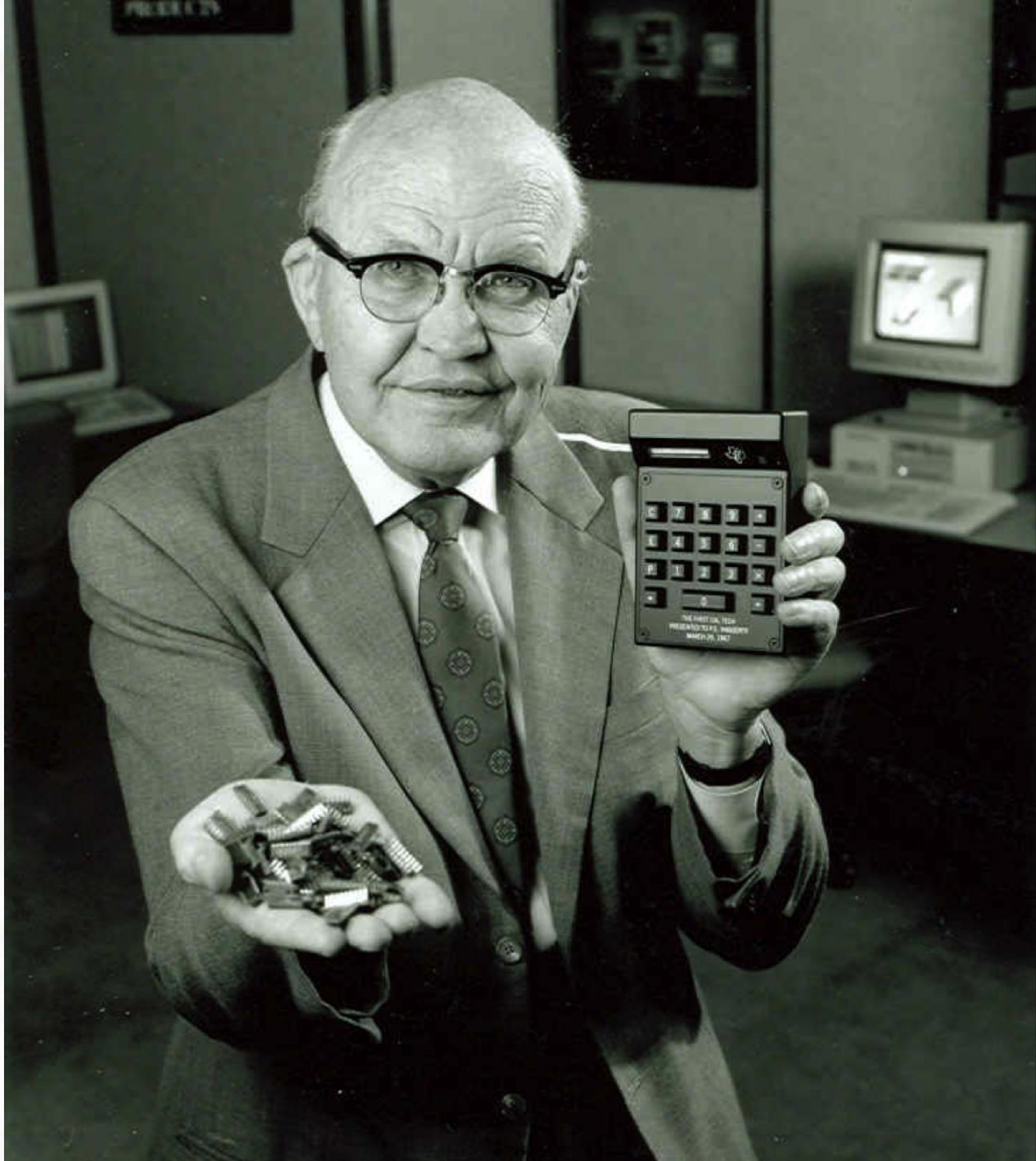
Bowmar was the first American company that made and sold their own line of portable electronic machines.

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.

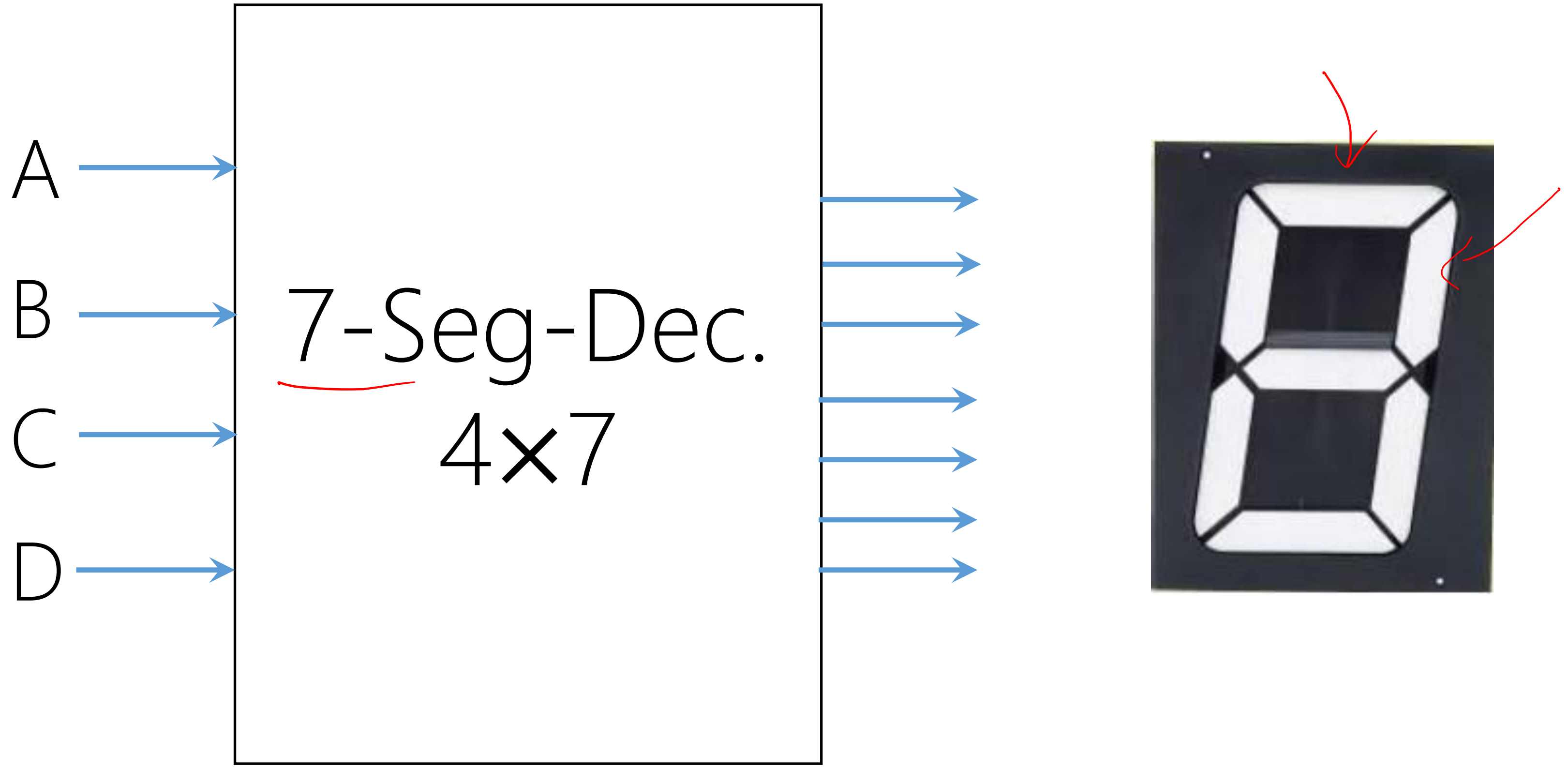
Bowmar wasn't too successful. The Japanese were using a flourescent style display that was cheaper and had a few design features the manufacturers liked better.

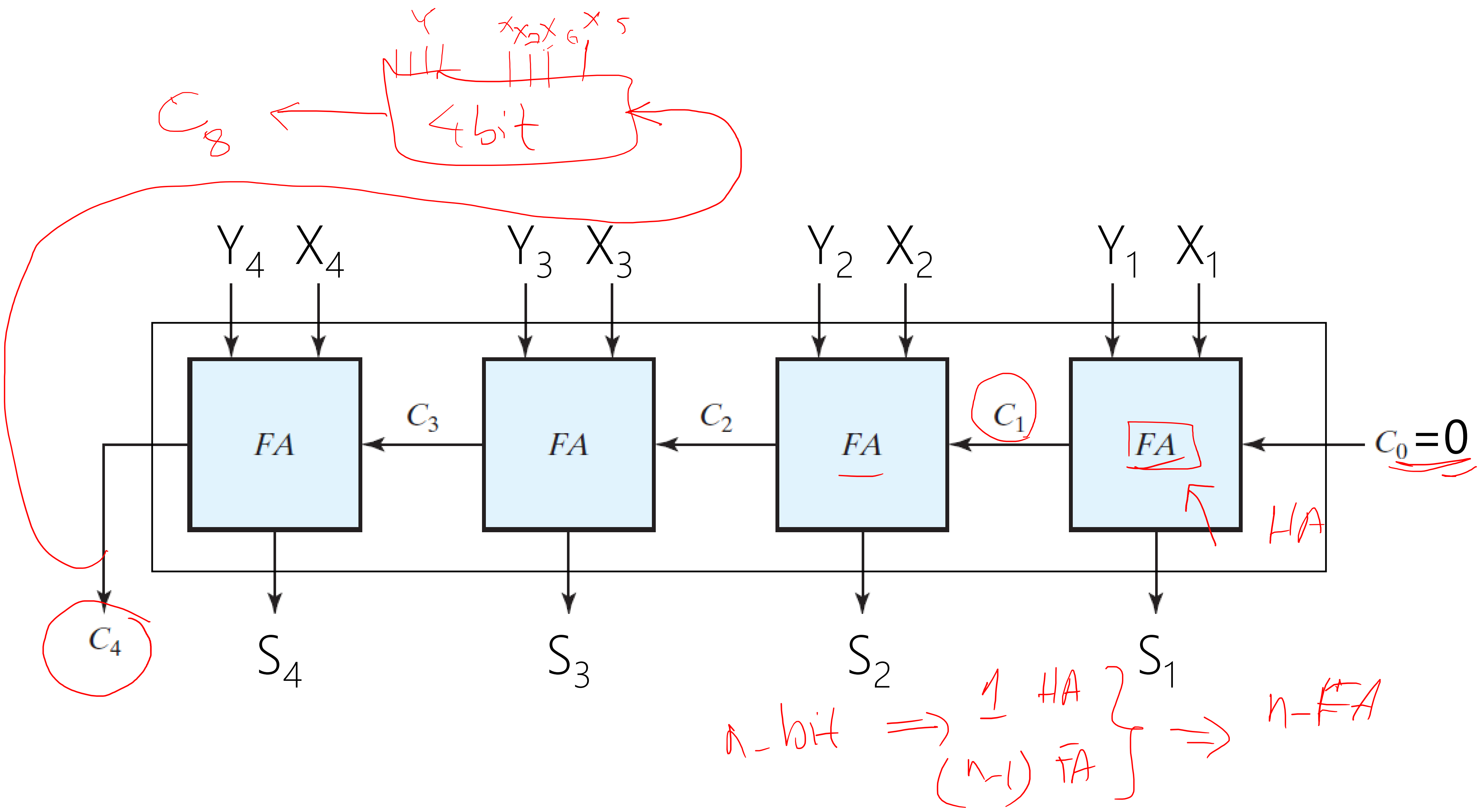
So, president Ed White, a consummate entrepreneur, and his staff came up with an even better idea — make the whole electronic calculator themselves.

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



Binary Number





Y X2 X 6 X 5
4bit
C8
Y4 X4
Y3 X3
Y2 X2
Y1 X1
FA
FA
FA
FA
C3
C2
C1
C0=0
HA
C4
S4
S3
S2
S1
n-bit => 1 HA
(n-1) FA => n-FA

0 X_1 1   0   1

$-$ 0 Y_1 $-$ 0 $-$ 1 $-$ 1

$B=0$ D_1 $B=0$ D_1 $\boxed{B}$ D_1 $B=0$ D_1

2-bits

3-bit

Binary Subtractor

Half-Subtractor ⟶ Full-Subtractor ⟶ n-bit Full-Subtractor

Lecture Assignments

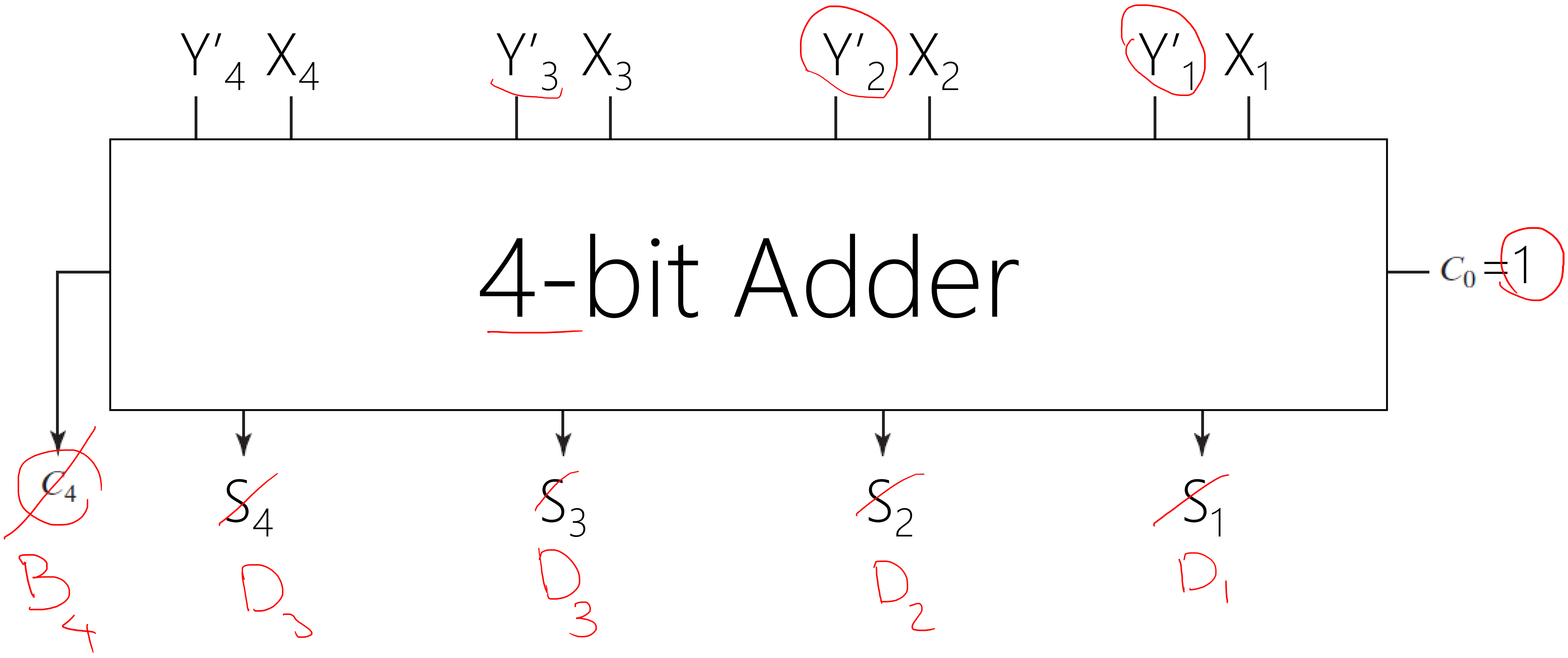
Binary Subtractor

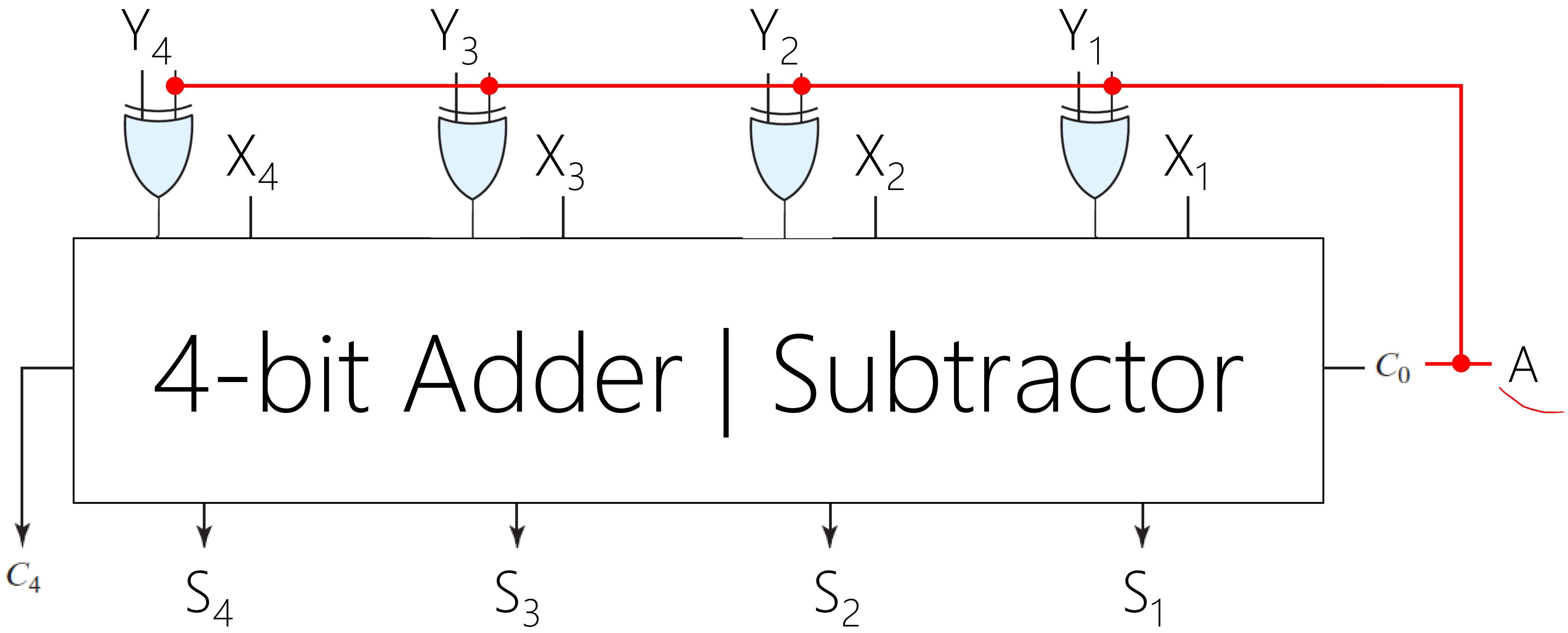
Signed-2’s-Complement

bitwise

$$X + Y' + (C_0=1)$$

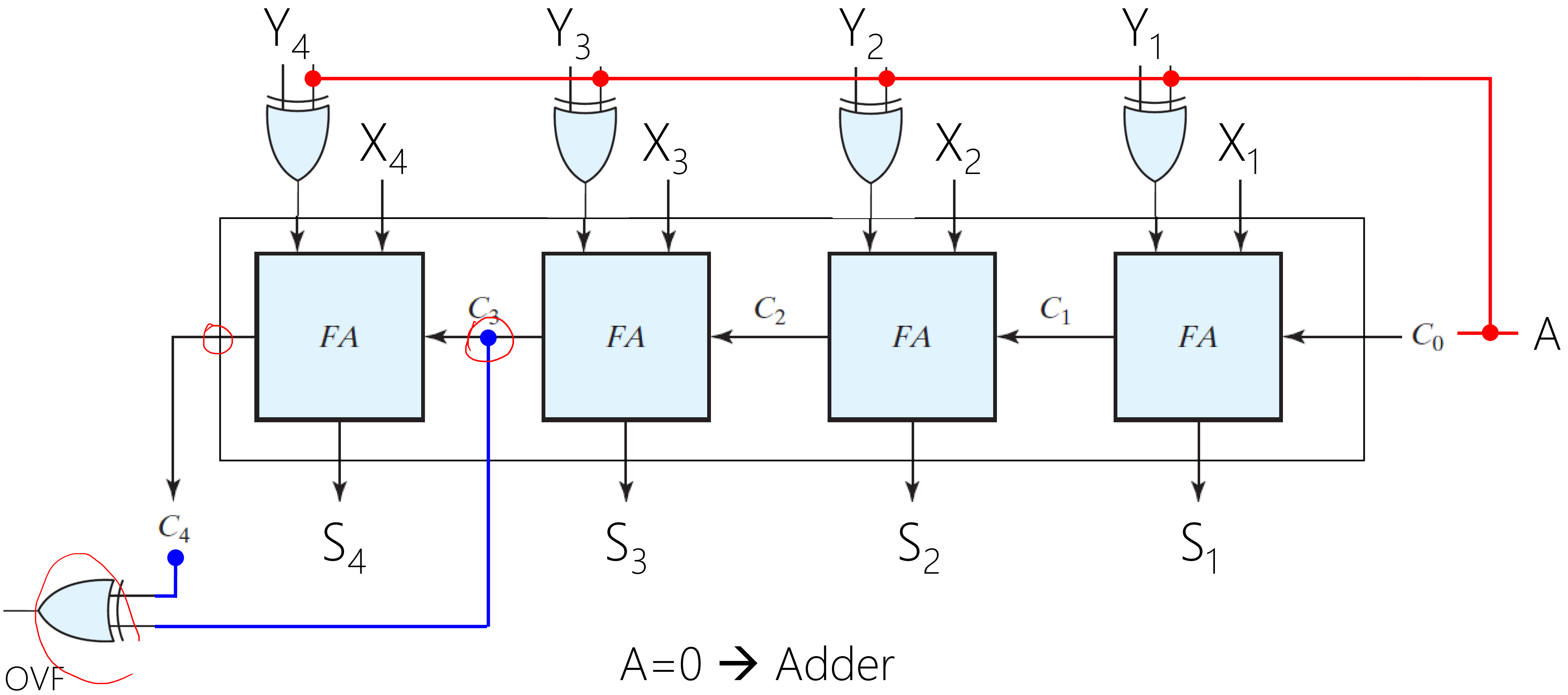
Subtraction in Signed-2's-Complement


Y'4 X4
Y'3 X3
Y'2 X2
Y'1 X1
4-bit Adder
C0 = 1
C4
S4
S3
S2
S1
B4
D
D3
D2
D1



A=0 → Adder

A=1 → Subtractor



A=0 → Adder
A=1 → Subtractor

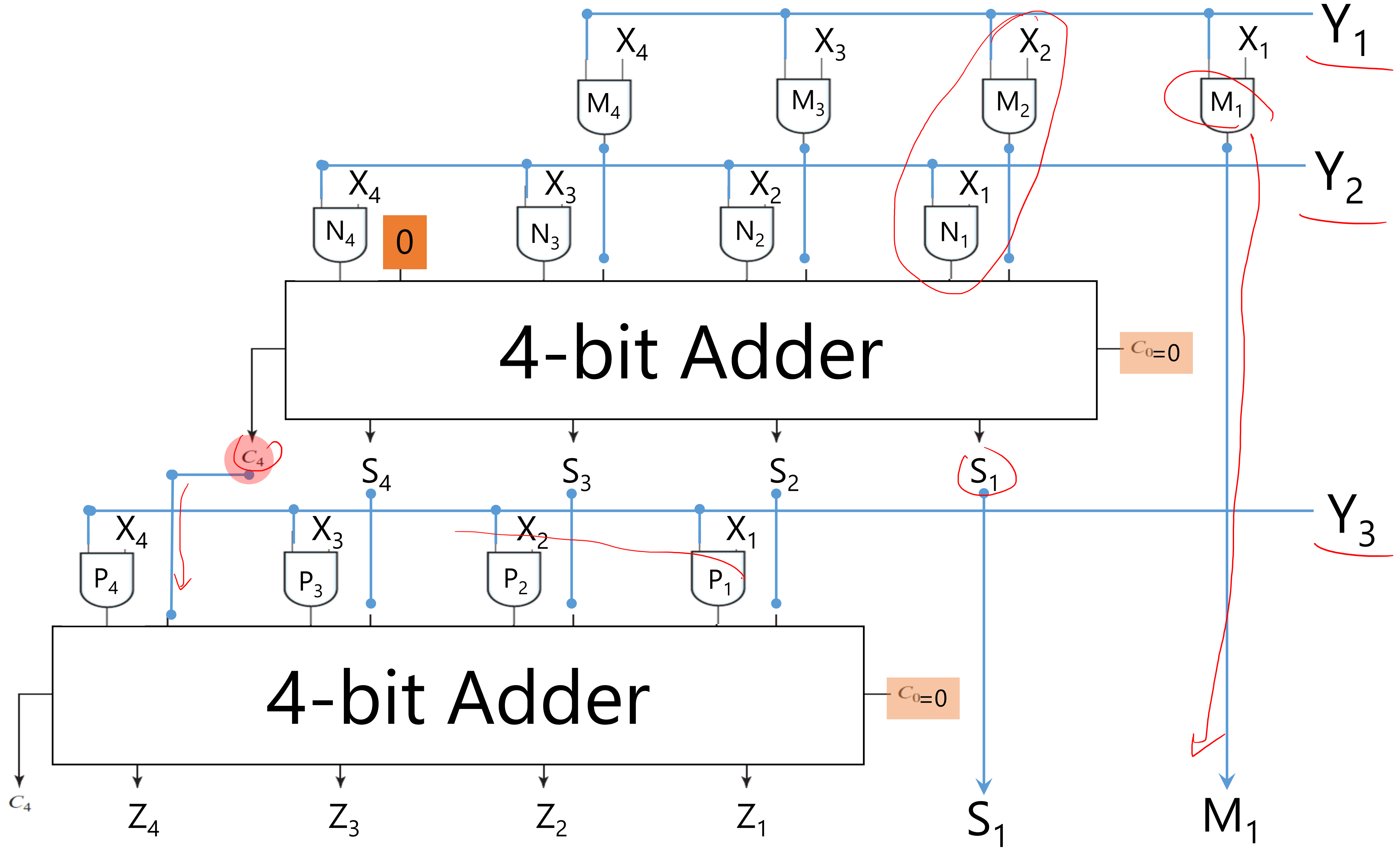
100
× 11

100 +
100 +
100 +

Binary Multiplier

Unsigned

$$\underbrace{\text{n-bit X} + \text{n-bit X} + \ldots + \text{n-bit X}}_{\text{m-bit Y times!}}$$



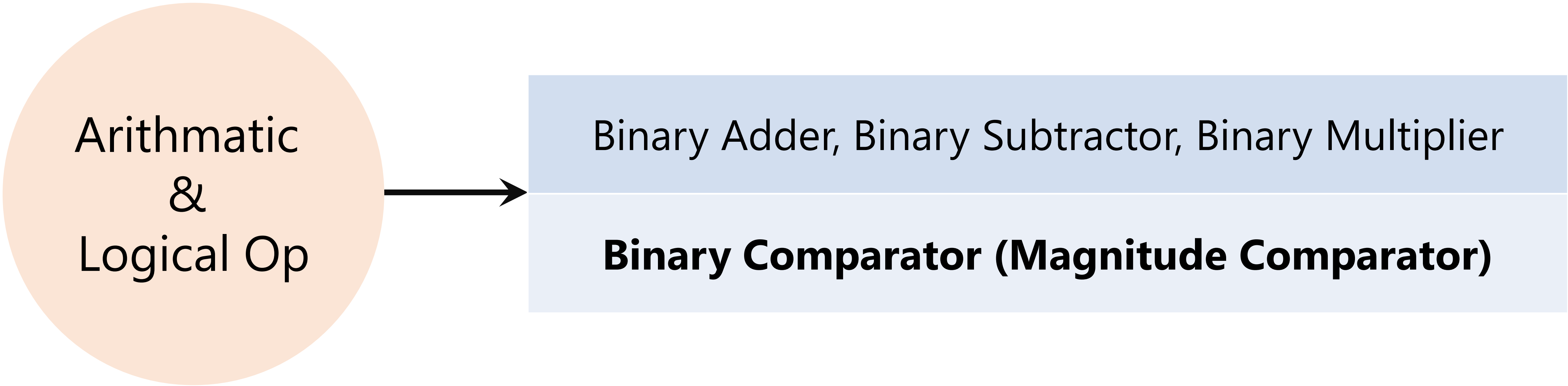
Y1
Y2
Y3
X4
X3
X2
X1
M4
M3
M2
M1
N4
N3
N2
N1
0
4-bit Adder
C0=0
C4
S4
S3
S2
S1
P4
P3
P2
P1
4-bit Adder
C0=0
C4
Z4
Z3
Z2
Z1
S1
M1

Binary Multiplier

Unsigned

n-bit X × m-bit Y

→ what is k in k-bit adders?


Arithmatic
&
Logical Op
Binary Adder, Binary Subtractor, Binary Multiplier
Binary Comparator (Magnitude Comparator)

Binary Comparator

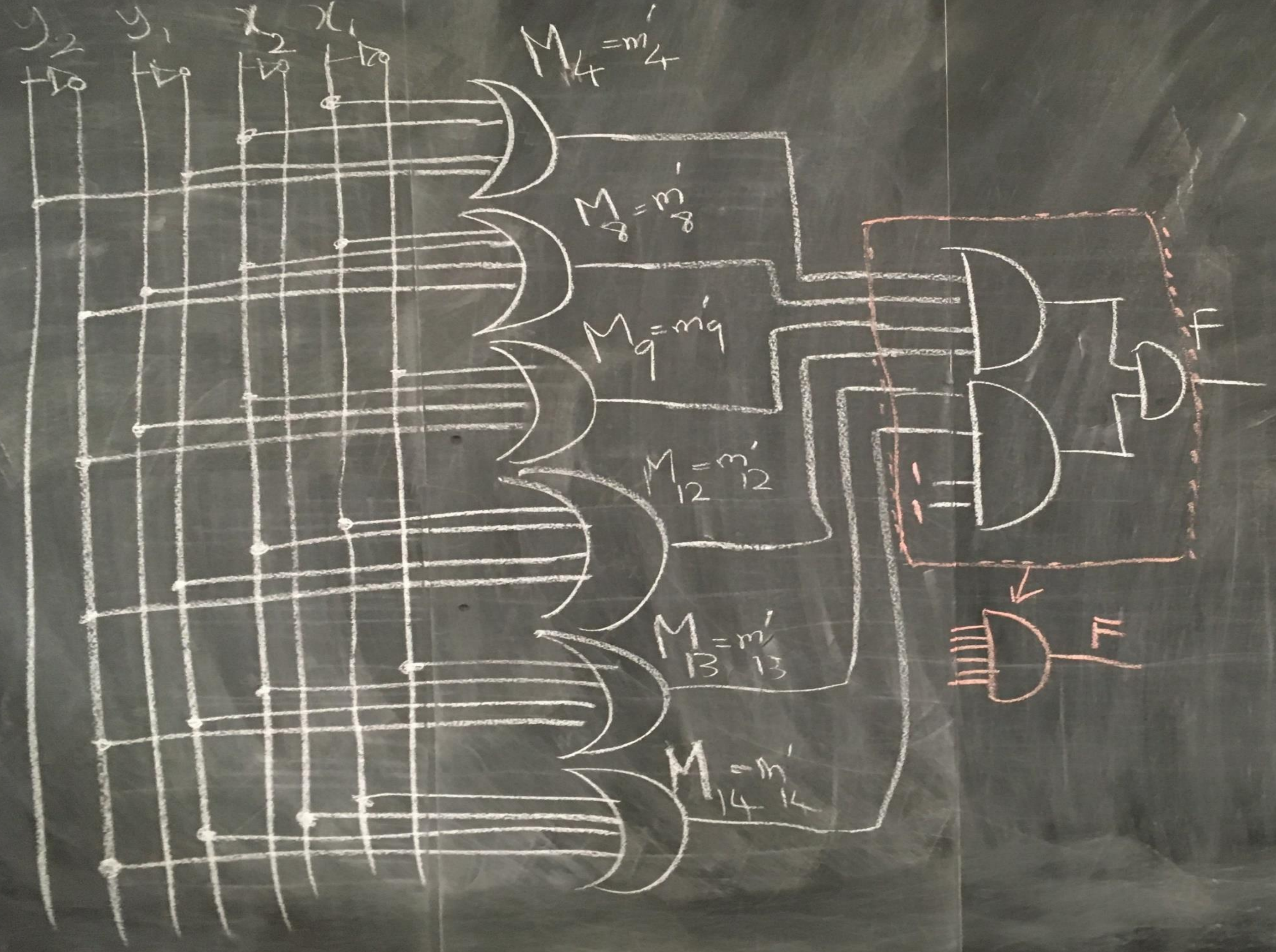
Unsigned

$X > Y$ $X == Y$ $X < Y$

Given two unsigned numbers x and y, design a logic circuit to see

$$x \overset{?}{\geq} y$$

Y2	Y1	X2	X1	$F(Y2,Y1,X2,X1)=\Sigma\ m(0,1,2,3,5,6,7,10,11,15)$	$F(Y2,Y1,X2,X1)=\Pi\ M(4,8,9,12,13,14)$
0	0	0	0	1	1
0	0	0	1	1	1
0	0	1	0	1	1
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	1	1



$$F = \prod M(4,8,9,12,13,14)$$

$$= M_4 M_8 M_9 M_{12} M_{13} M_{14}$$

$$= m_4' m_8' m_9' m_{12}' m_{13}' m_{14}'$$

$$= (y_2' y_1 x_2' x_1')' \quad m_4' \rightarrow y_2 + y_1' + x_2 + x_1$$

$$(y_2 y_1' x_2' x_1')' \quad m_8' \rightarrow y_2' + y_1 + x_2 + x_1$$

$$(y_2 y_1' x_2' x_1)' \quad m_9' \rightarrow y_2' + y_1 + x_2 + x_1'$$

$$(y_2 y_1 x_2' x_1')' \quad m_{12}' \rightarrow y_2' + y_1' + x_2 + x_1$$

$$(y_2 y_1 x_2' x_1)' \quad m_{13}' \rightarrow y_2' + y_1' + x_2 + x_1'$$

$$(y_2 y_1 x_2 x_1')' \quad m_{14}' \rightarrow y_2' + y_1' + x_2' + x_1$$

Given two unsigned numbers x and y, design a logic circuit to see

$$x > y\,; x == y\,;\ x < y$$

Y2	Y1	X2	X1	$F_1 = (X > Y)$	$F_2 = (X == Y)$	$F_3 = (X < Y)$
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

Y2	Y1	X2	X1	$F_1 = (X > Y)$	$F_2 = (X == Y)$	$F_3 = (X < Y)$
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1		0
0	0	1	1	1		0
0	1	0	0	0		1
0	1	0	1	0		0
0	1	1	0	1		0
0	1	1	1	1		0
1	0	0	0	0		1
1	0	0	1	0		1
1	0	1	0	0		0
1	0	1	1	1		0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

If X and Y
3, 4, 5, …
bits?!

Binary Subtractor?

$0 \odot 1 = 0$ $\quad$ $1 \odot 1 = 1$

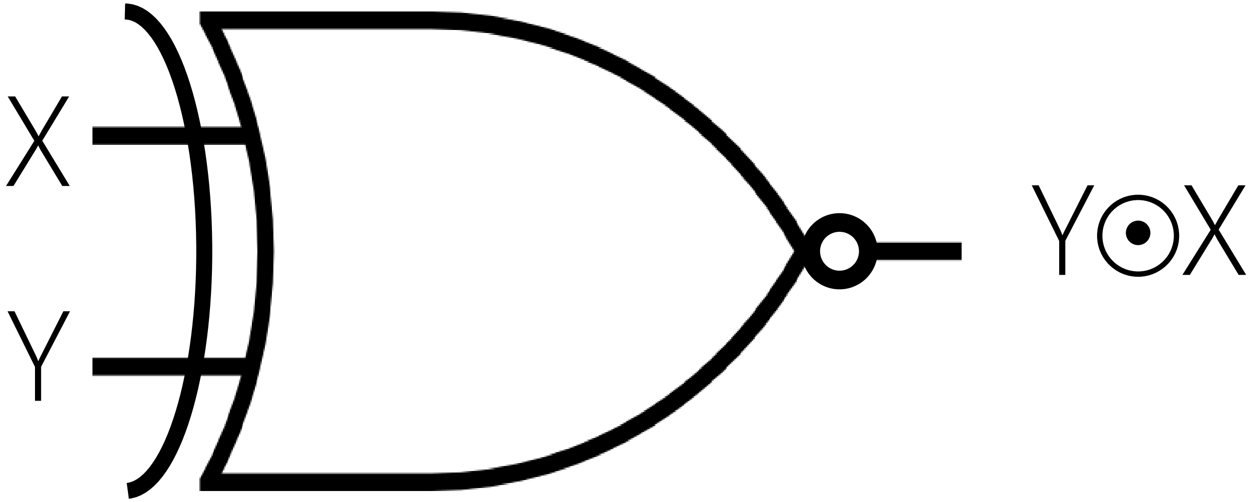
$1 \odot 0 = 0$ $\quad$ $0 \odot 0 = 1$

XNOR

Equality Gate

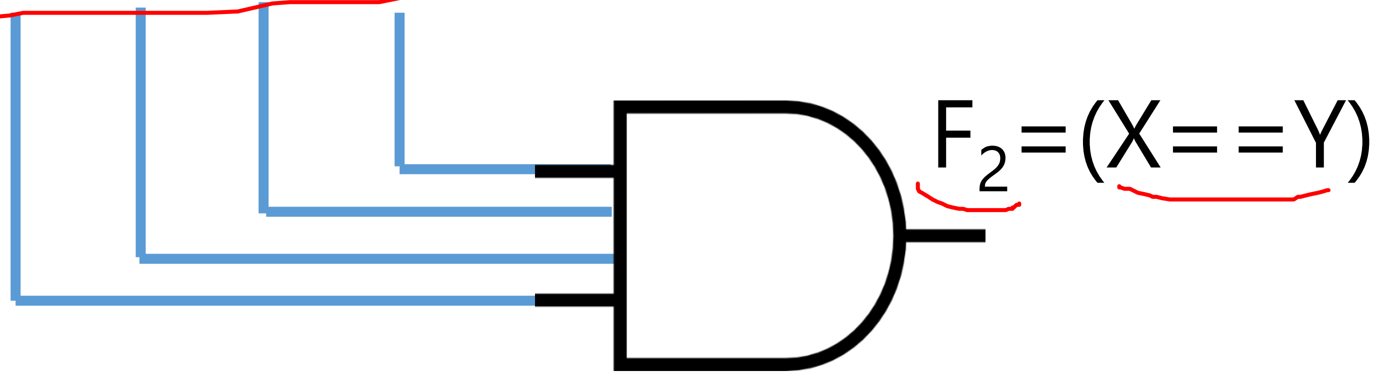
NOT Exclusive-OR (XNOR)

W04



Y	X	$F = F(Y,X) = Y'X'+YX = m_0+m_3$
0	0	1
0	1	0
1	0	0
1	1	1

$$
\begin{array}{r}
X_4X_3X_2X_1 \\
\odot \quad Y_4Y_3Y_2Y_1 \\
\hline
1\;\;1\;\;1\;\;1
\end{array}
$$



$r^{n-1} \ldots r^1 \quad r^0 \quad r$

unsign

$$X_4 = 1\ \cancel{X_3 X_2 X_1}$$

$$Y_4 = 0\ \cancel{Y_3 Y_2 Y_1}$$

$$X_4 Y'_4 \rightarrow X > Y$$

$$X_4 = 0\ X_3 X_2 X_1$$

$$Y_4 = 1\ Y_3 Y_2 Y_1$$

$$X'_4 Y_4 \rightarrow X < Y$$

$$
\begin{array}{r}
X_4\,X_3X_2X_1 \\
Y_4\,Y_3Y_2Y_1 \\
\hline
X_4 \odot Y_4 = 1
\end{array}
$$

$$
\begin{array}{r}
X_4\ X_3{=}1\ X_2X_1 \\
Y_4\ Y_3{=}0\ Y_2Y_1 \\
\hline
\end{array}
$$

$$X_4 \odot Y_4 = 1$$

$$X_3Y'_3 \rightarrow X > Y$$

$$
\begin{array}{r}
X_4\ X_3=0\ X_2X_1 \\
Y_4\ Y_3=1\ Y_2Y_1 \\
\hline
\end{array}
$$

$$X_4 \odot Y_4 = 1$$

$$X'_3Y_3 \rightarrow X < Y$$

$$
\begin{aligned}
F1 = (X > Y) = \; & X_4 Y'_4 + \\
& (X_4 \odot Y_4) X_3 Y'_3 + \\
& (X_4 \odot Y_4)(X_3 \odot Y_3) X_2 Y'_2 + \\
& (X_4 \odot Y_4)(X_3 \odot Y_3)(X_2 \odot Y_2) X_1 Y'_1
\end{aligned}
$$

$$F1 = (X < Y) = X'_4 Y_4 + (X_4 \odot Y_4) X'_3 Y_3 + (X_4 \odot Y_4)(X_3 \odot Y_3) X'_2 Y_2 + (X_4 \odot Y_4)(X_3 \odot Y_3)(X_2 \odot Y_2) X'_1 Y_1$$

m

SOP

$x \odot y$

$xy + x'y'$

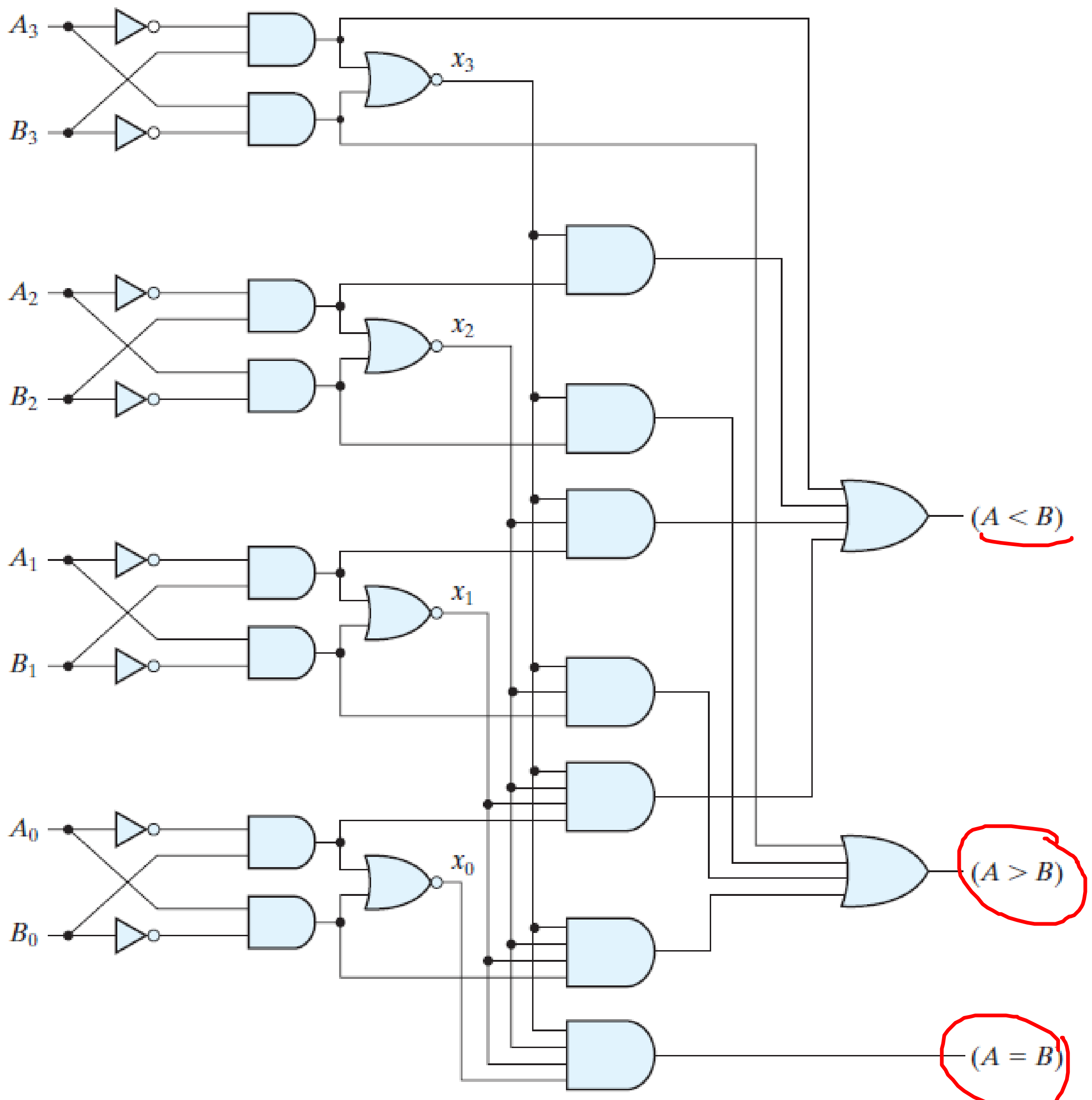
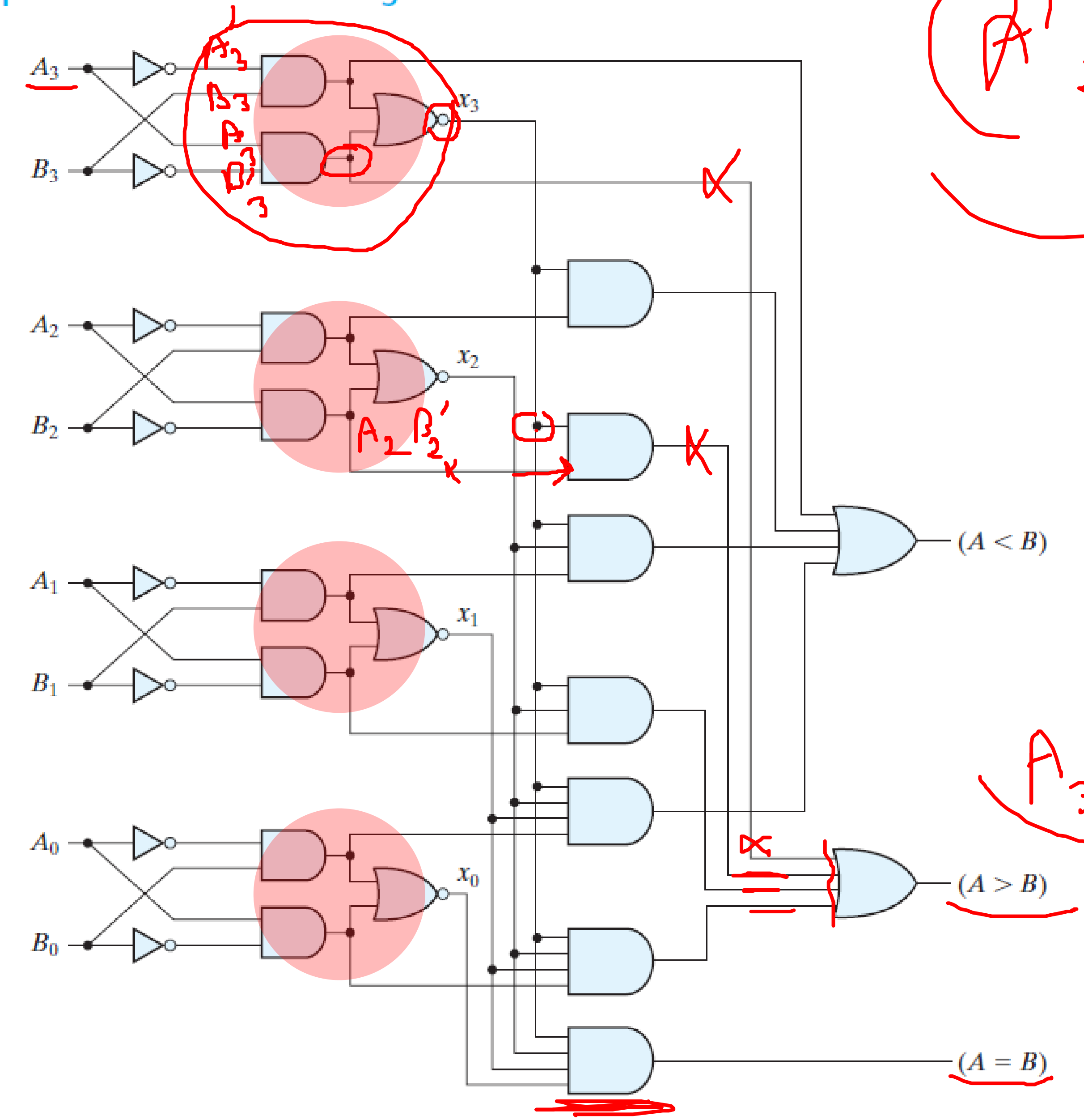


FIGURE 4.17
Four-bit magnitude comparator

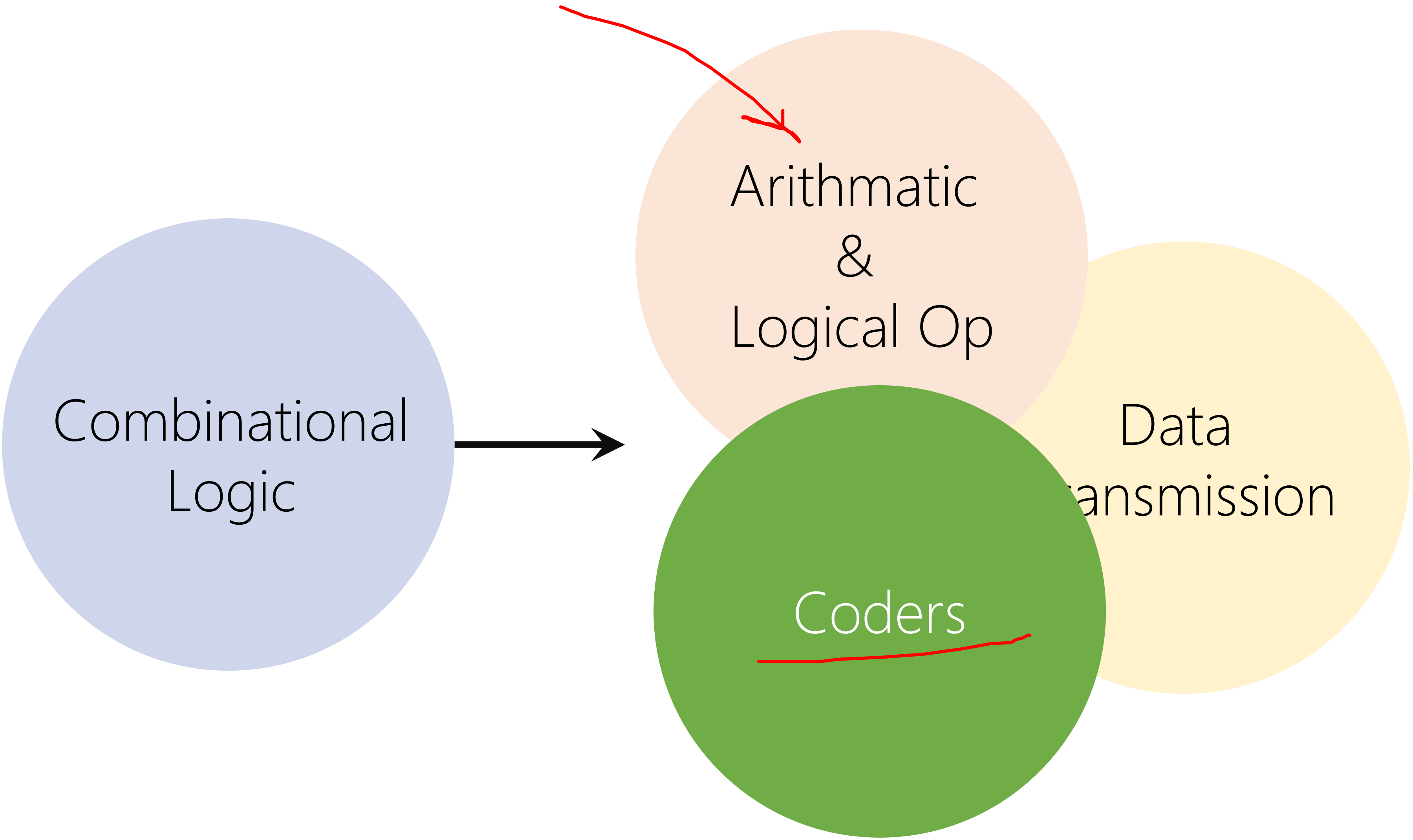
$$\left(A'_3 B_3 + A_3 B'_3\right)'$$

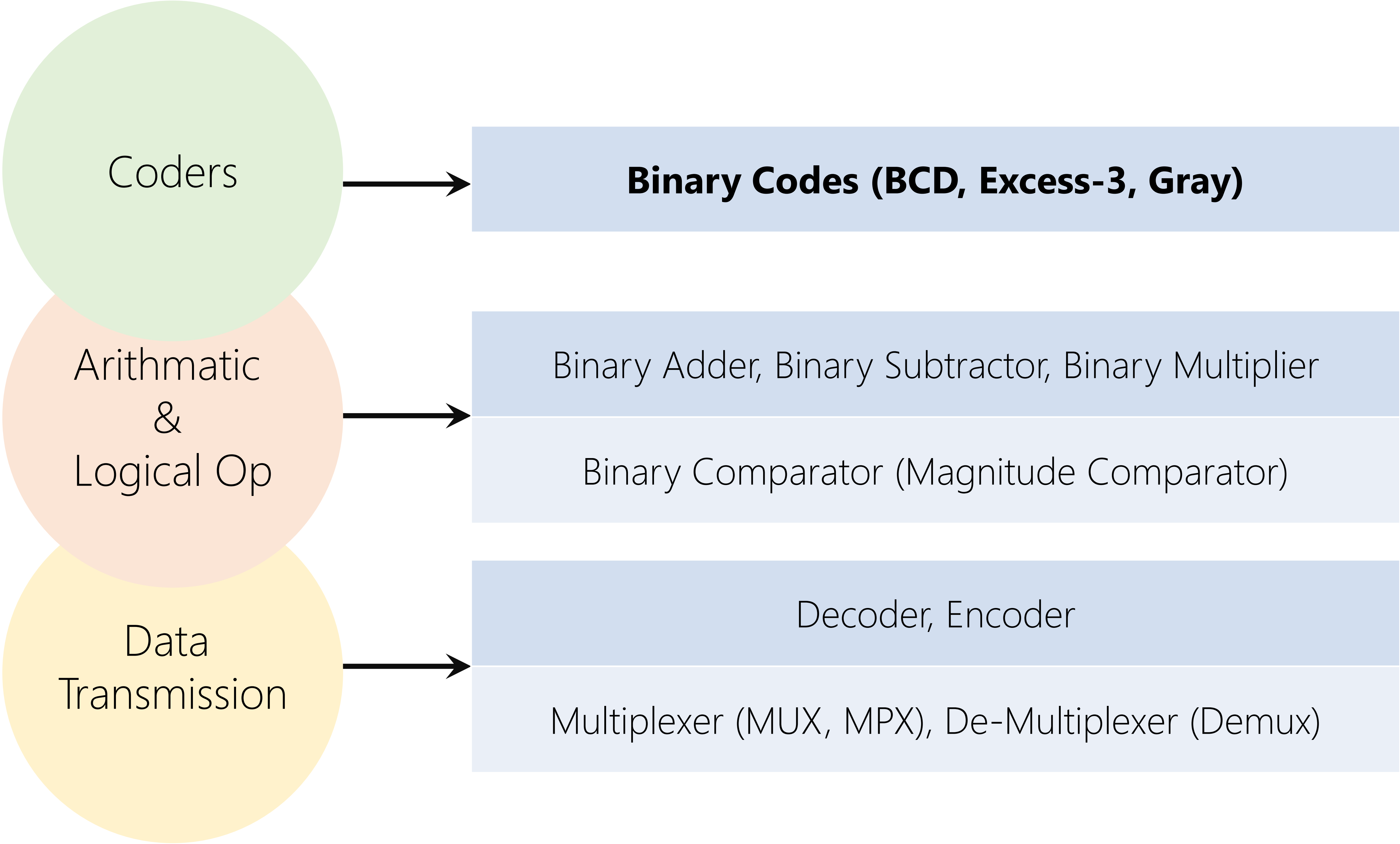


$$A_3 B'_3$$

$$(A_3 \odot B_3)\, A_2 B'_2$$

FIGURE 4.17
Four-bit magnitude comparator


Combinational
Logic
Arithmatic
&
Logical Op
Data
ansmission
Coders


Coders
Binary Codes (BCD, Excess-3, Gray)
Arithmatic & Logical Op
Binary Adder, Binary Subtractor, Binary Multiplier
Binary Comparator (Magnitude Comparator)
Data Transmission
Decoder, Encoder
Multiplexer (MUX, MPX), De-Multiplexer (Demux)

Coding

A → Encode→B

A ← Decode ← B

A ↔ [Enc][Dec]code ↔ B

$(12)_{10} \rightleftarrows (11..)_2$

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 → Encode → B

A $\nleftarrow$ Decode $\nleftarrow$ B

2-way Coding

A ↔ Look up Table ↔ B

Base-64

A ↔ Look up Table ↔ 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

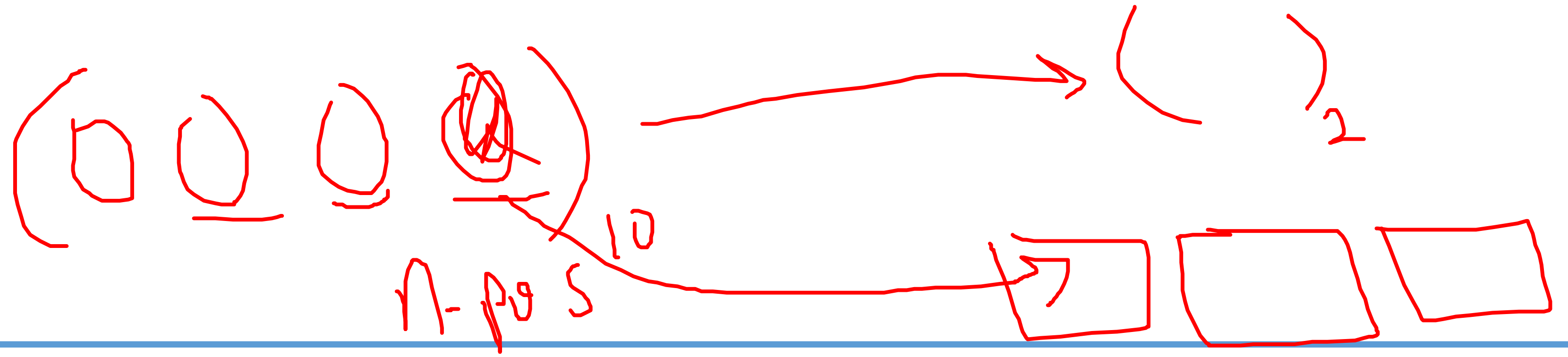
Assigning binary numbers to things

A ↔ Look up Table ↔ Binary Code

Binary Codes

Not Necessarily follow Radix Number System

A ↔ Look up Table ↔ Binary Code



Binary Coded Decimal

BCD (8421)

Decimal ↔ Look up Table ↔ Binary Code

Table 1.4
Binary-Coded Decimal (BCD)

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

10 0001 0000

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
...	...	...

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

$$(185)_{10} = (0001)_{BCD} = (?)_2$$

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

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

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

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

	Remainder
$185 \div 2$	1
$92 \div 2$	0
$46 \div 2$	0
$23 \div 2$	1
$11 \div 2$	1
$5 \div 2$	1
$2 \div 2$	0
$1 \div 2$	1
0	

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

Other Binary Codes

A ↔ Look up Table ↔ B

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

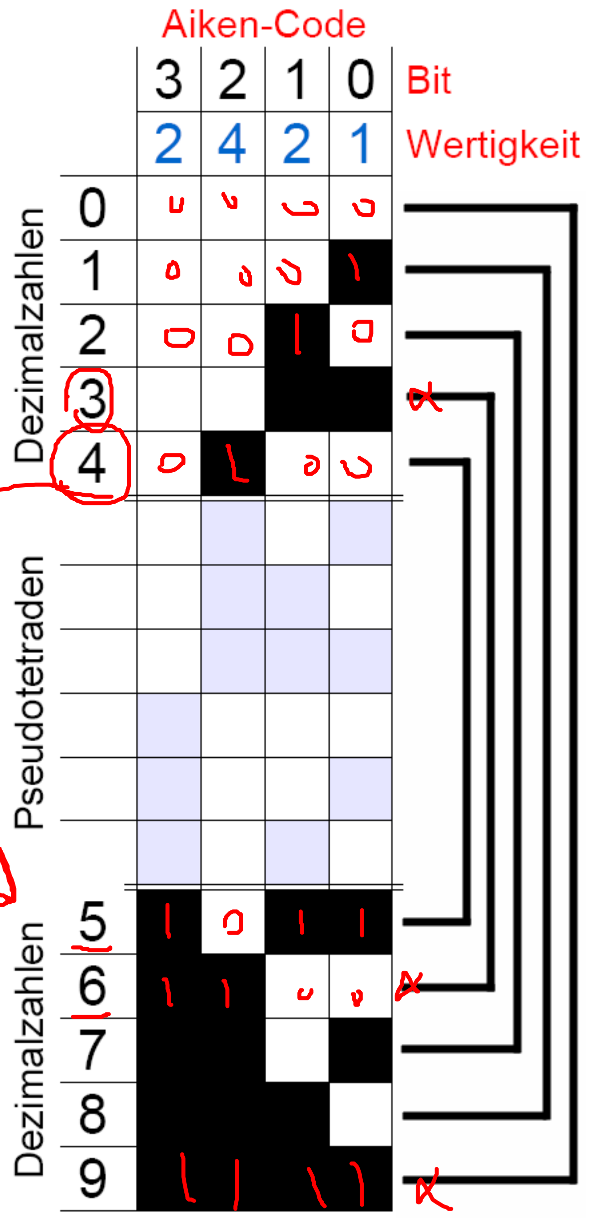
Other Binary Codes

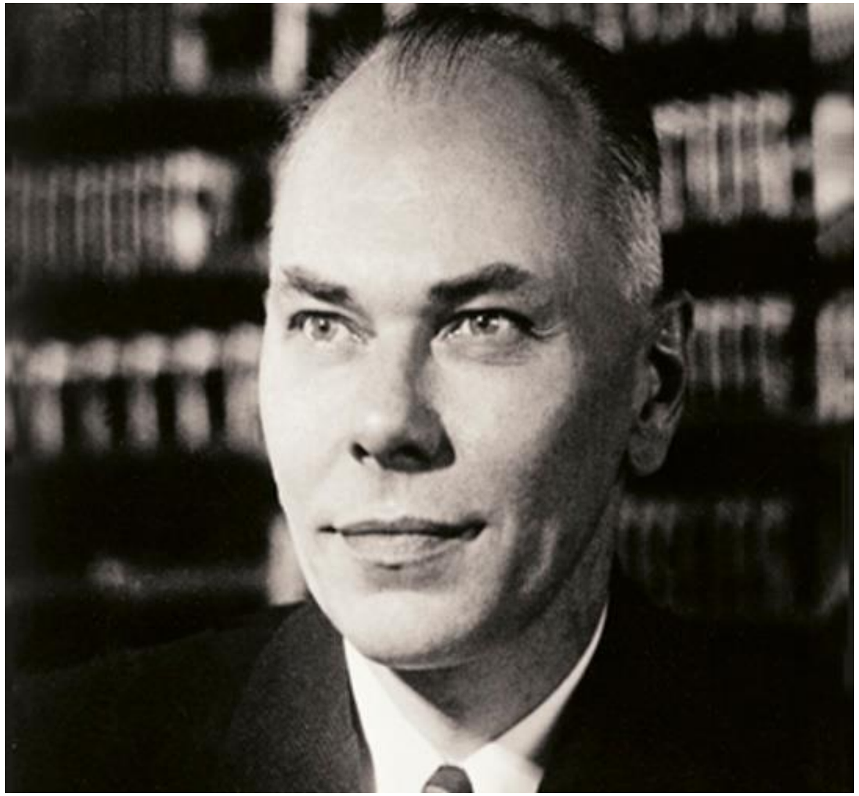
Aiken (2421)

/ˈeɪkən/

2^{0} 2^{2} 2^{1} 2^{0}

https://en.wikipedia.org/wiki/Aiken_code





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

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

NOT

9's (9) = 0   1's (0000)

1111

After 4, NOT of 9's Comp!

$$
\begin{aligned}
(185)_{10} &= (0001\ 1000\ 0101)_{BCD\ (8421)} \\
&= (10111001)_{2} \\
&= (0001\ \text{NOT}(9-8)\ \text{NOT}(9-5))_{Aiken\ (2421)}
\end{aligned}
$$

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

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

$$(185)_{10} = (0001\ 1000\ 0101)_{\mathrm{BCD}\ (8421)}$$

$$= (10111001)_2$$

$$= (0001\ 1110\ 1011)_{\mathrm{Aiken}\ (2421)}$$

1011

1×2^1

1×2^0

1×2^1

0×2^2

$2 + 1 + 2 + 0 = 5$

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	Aiken 2421	+3 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

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

$$
\begin{aligned}
(185)_{10} &= (0001\ 1000\ 0101)_{BCD\ (8421)} \\
&= (10111001)_2 \\
&= (0001\ 1110\ 1011)_{Aiken\ (2421)} \\
&= ((4)\ (11)\ (8))_{Excess\text{-}3}
\end{aligned}
$$

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

Other Binary Codes

84(-2)(-1)

2^3 2^2

2 2

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

1x−1 +1x−2
1x4
0x8
1

NOT

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

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

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

999

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

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

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

Other Binary Codes
Gray

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

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

1-bit change

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.

$(20)_{10}$	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**.

$(20)_{10}$	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.

$(20)_{10}$	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.

$(20)_{10}$	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.

$(20)_{10}$	Binary Number	1	0	1	0	0
	Gray Code	1	1	1	$1 \oplus 0 = 1$	x

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

$(20)_{10}$	Binary Number	1	0	1	0	0
	Gray Code	1	1	1	1	$0 \oplus 0 = 0$

$(21)_{10}$

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

$(20)_{10}$	Binary Number	1	0	1	0	0
	Gray Code	1	1	1	1	0

$(21)_{10}$	Binary Number	1	0	1	0	1
	Gray Code	1	1	1	1	1

ASCII Code

American **S**tandard **C**ode for **I**nformation **I**nterchange

USASCII code chart

0 11 00 00

b_7 b_6 b_5 →					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4	b_3	b_2	b_1	Row ↓ / Column →	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

010 0101

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

Table 1.7
American Standard Code for Information Interchange (ASCII)

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	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	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					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

Combinational Logic

Code Conversion

BCD (8421) → Excess-3

Table 4.2
Truth Table for Code Conversion Example

Input BCD				Output Excess-3 Code			
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

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	? X	? X	?	?
1	0	1	1	? X	? X	?	?
1	1	0	0	? X	? X	?	?
1	1	0	1	? X	?	?	?
1	1	1	0	? X	?	?	?
1	1	1	1	? X	?	?	?

Don’t Care Conditions

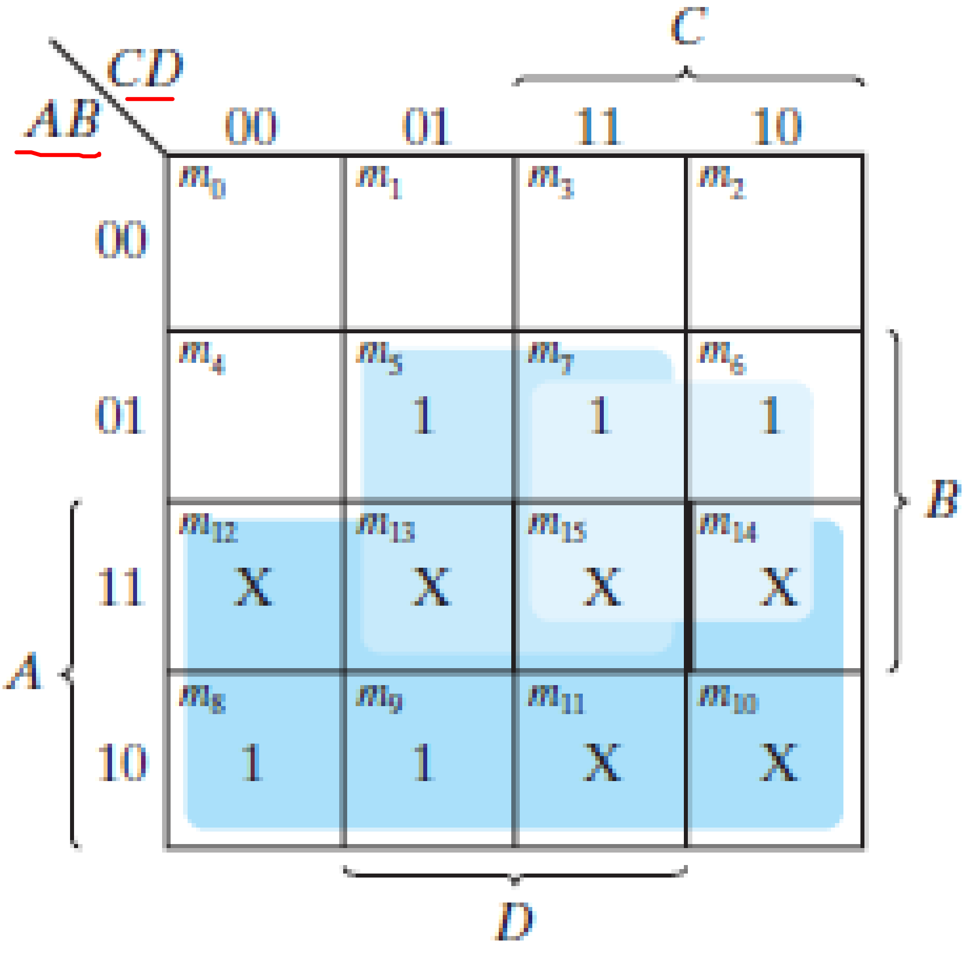
In practice, in some applications the function is not specified for certain combinations of the variables.

Don’t Care Conditions

Functions that have unspecified outputs for some input combinations are called *incompletely specified functions*.

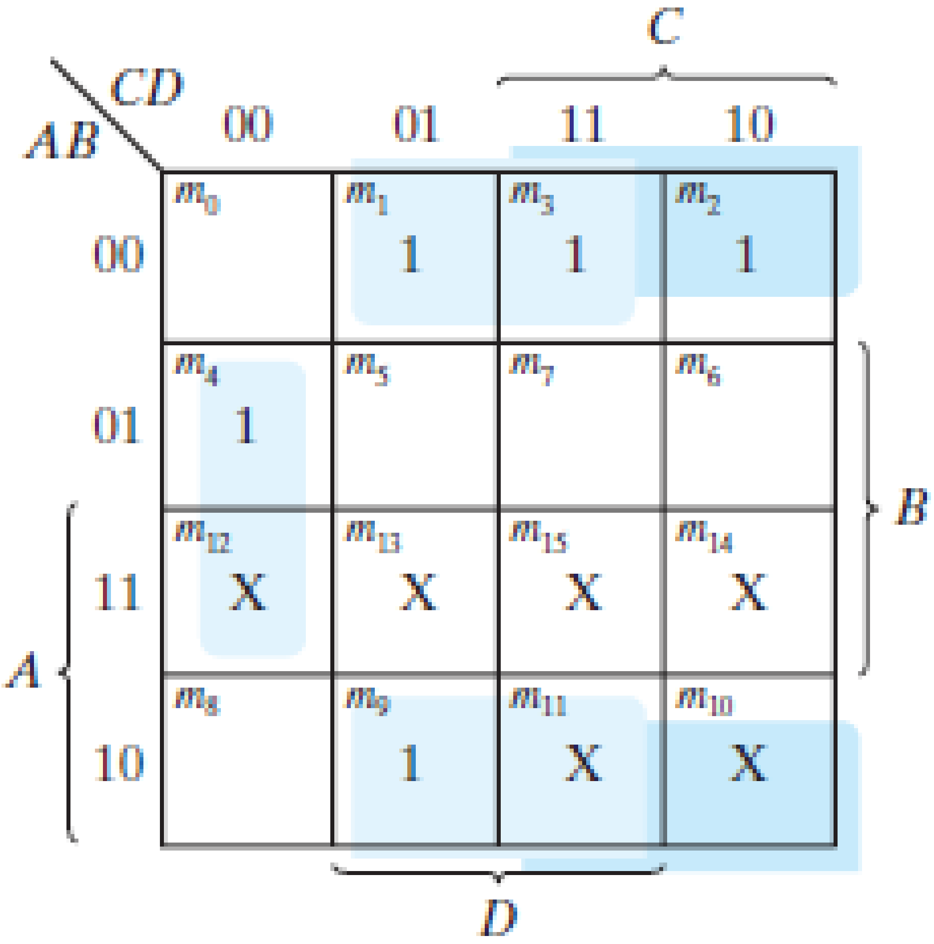
Don’t-care conditions can be used on a map to provide further simplification of the Boolean expression.

$$W(A,B,C,D) = \sum(5,6,7,8,9) + d(10,11,12,13,14,15)$$



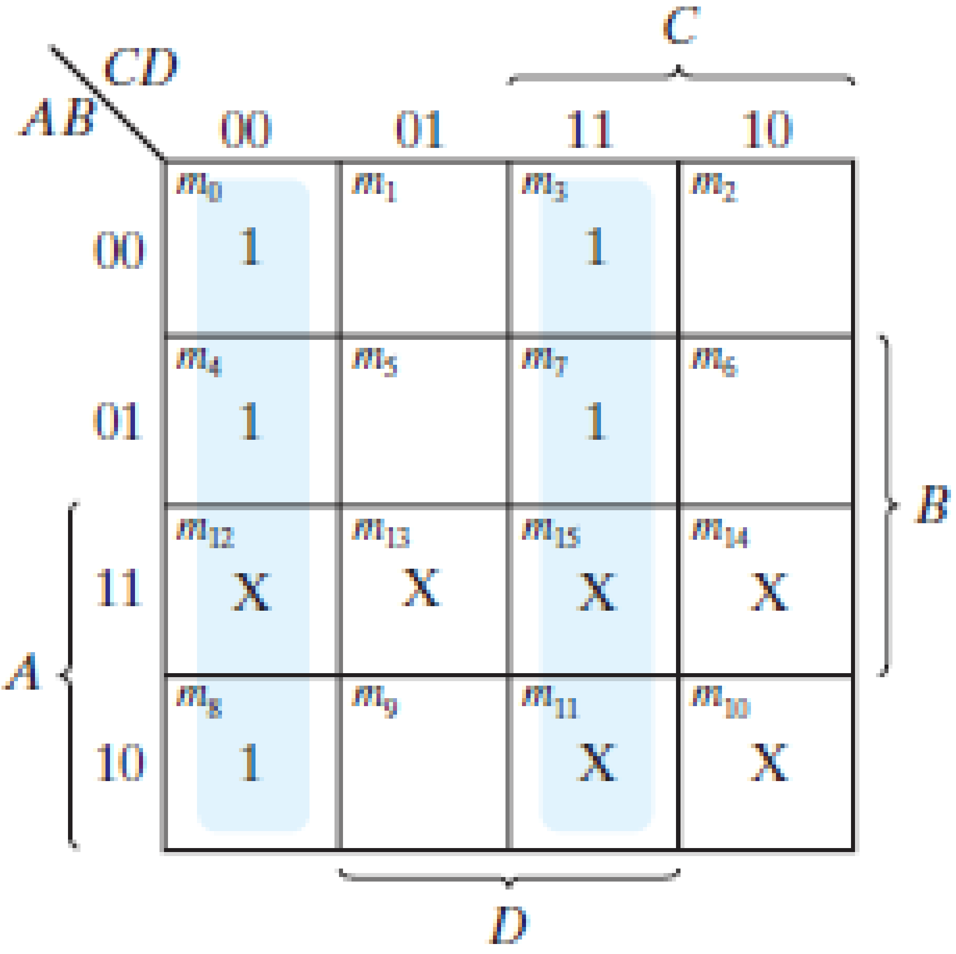
$w = A + BC + BD$

$X(A,B,C,D) = \sum(1,2,3,4,9) + d(10,11,12,13,14,15)$



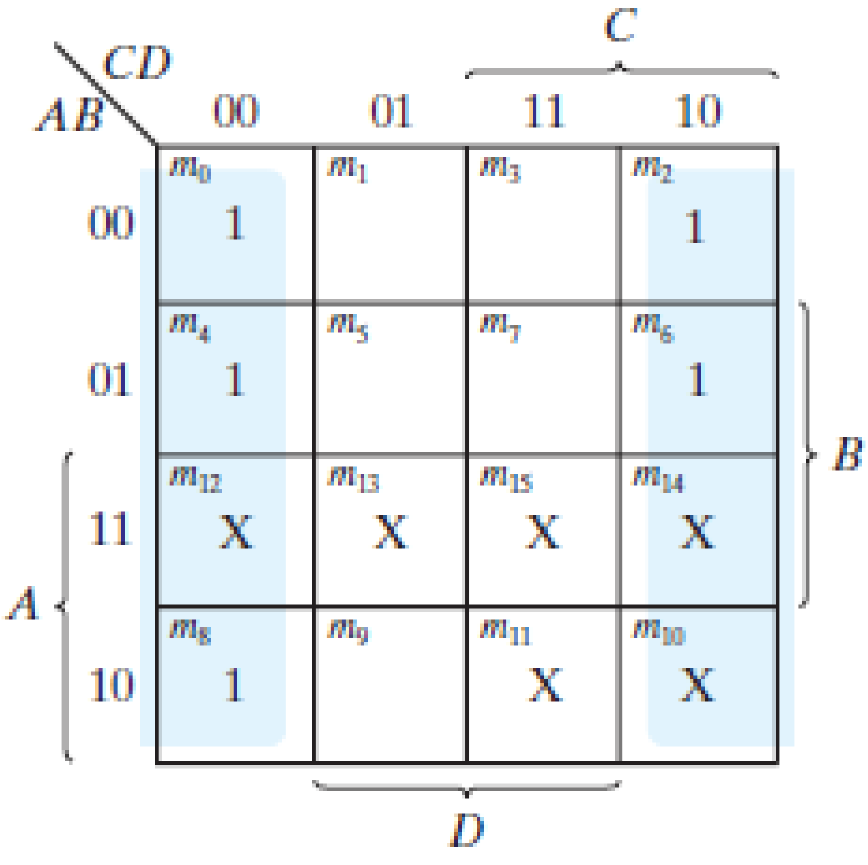
$x = B'C + B'D + BC'D'$

$$Y(A,B,C,D) = \sum(0,3,4,7,8) + d(10,11,12,13,14,15)$$



$y = CD + C'D'$

$Z(A,B,C,D) = \sum(0,2,4,6,8) + d(10,11,12,13,14,15)$



$z = D'$

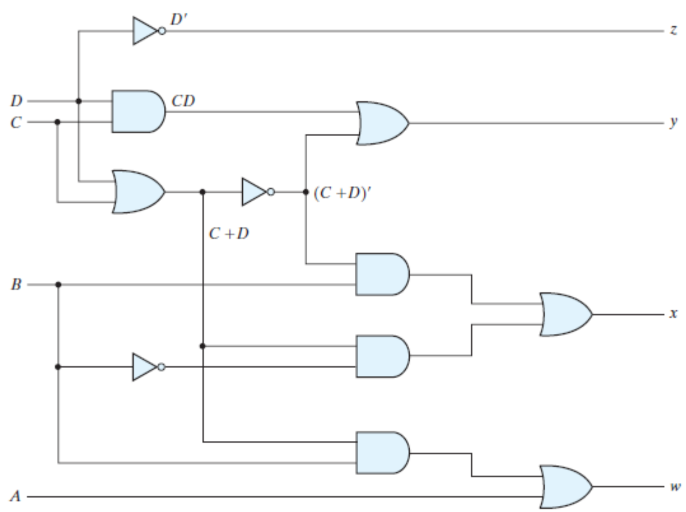
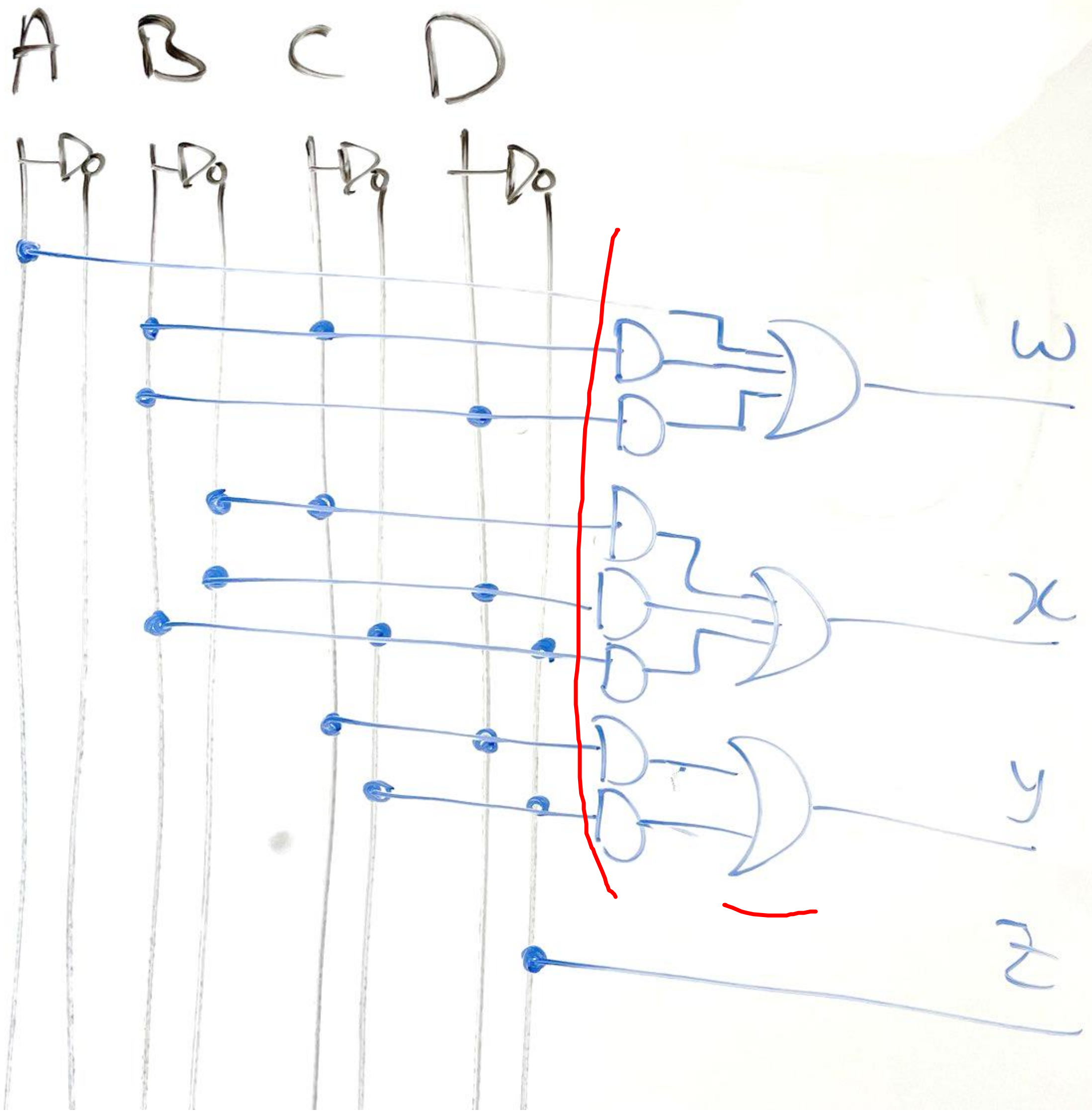


FIGURE 4.4
Logic diagram for BCD-to-excess-3 code converter



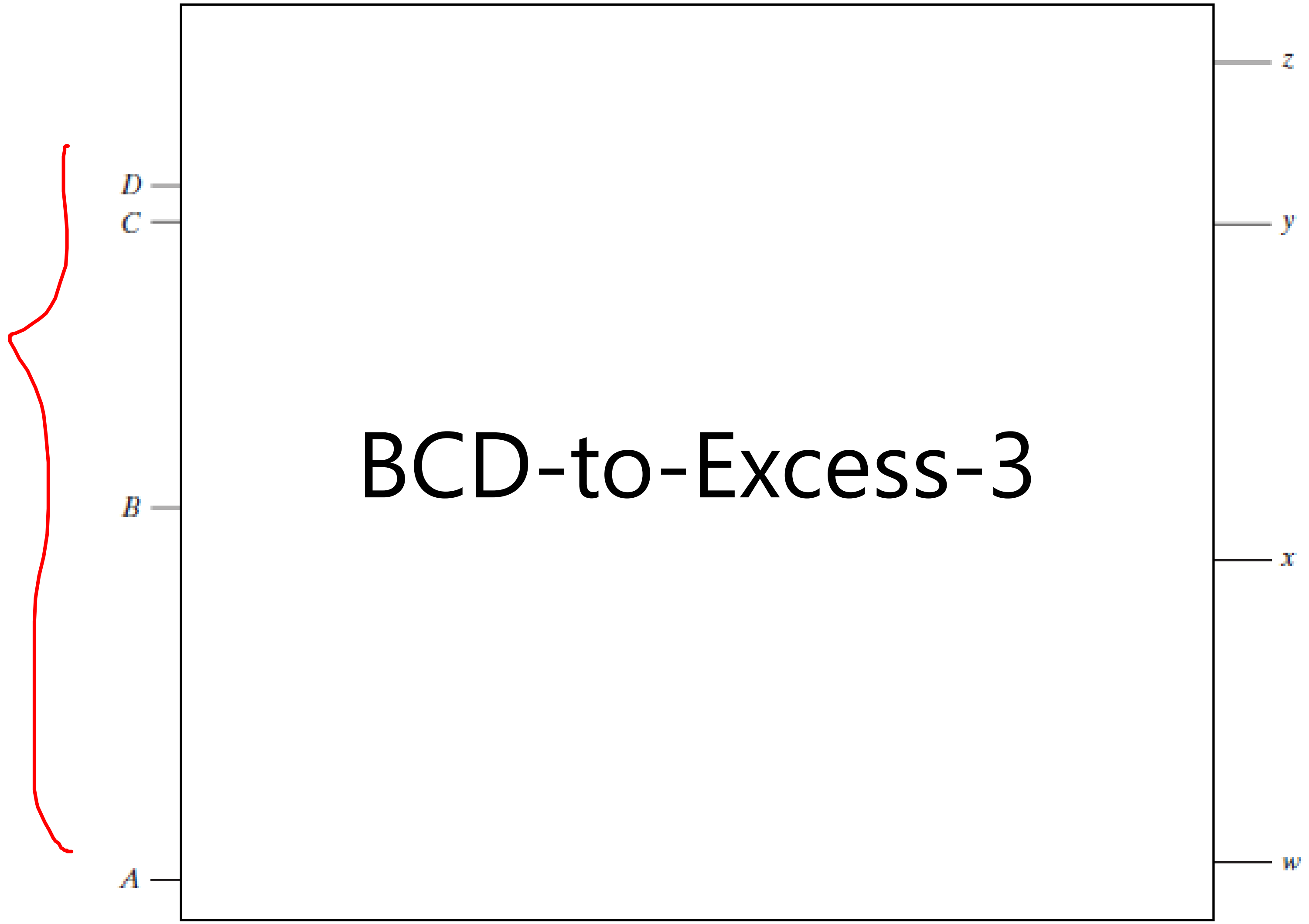


FIGURE 4.4
Logic diagram for BCD-to-excess-3 code converter

MAXTERM

$$
\begin{aligned}
W(A,B,C,D) &= \sum(5,6,7,8,9) + d(10,11,12,13,14,15) \\
&= \prod(?)
\end{aligned}
$$

$$
\begin{aligned}
W(A,B,C,D) &= \sum(5,6,7,8,9) + d(10,11,12,13,14,15) \\
&= \prod(0,1,2,3,4)
\end{aligned}
$$

$$W(A,B,C,D) = \sum(5,6,7,8,9) + d(10,11,12,13,14,15)$$
$$= \prod(0,1,2,3,4) + D(10,11,12,13,14,15)$$

We can assume the don't care conditions are 0 if they help to more simplification

$$
\begin{aligned}
W(A,B,C,D) &= \sum(5,6,7,8,9) + d(10,11,12,13,14,15) \\
&= \prod(0,1,2,3,4) + D(10,11,12,13,14,15) \\
&= ()'
\end{aligned}
$$

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

$$
\begin{aligned}
W(A,B,C,D) &= \sum(5,6,7,8,9) + d(10,11,12,13,14,15) \\
&= \prod(0,1,2,3,4) + D(10,11,12,13,14,15) \\
&= ((A'B'))'
\end{aligned}
$$

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

$$W(A,B,C,D) = \sum(5,6,7,8,9) + d(10,11,12,13,14,15)$$
$$= \prod(0,1,2,3,4) + D(10,11,12,13,14,15)$$
$$= ((A'B')+(A'C'D'))'$$

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

$$W(A,B,C,D) = \sum(5,6,7,8,9) + d(10,11,12,13,14,15)$$
$$= \prod(0,1,2,3,4) + D(10,11,12,13,14,15)$$
$$= ((A'B')+(A'C'D'))'$$ Here the "don't care conditions" did not help ☹
$$= (A+B)(A+C+D)$$

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

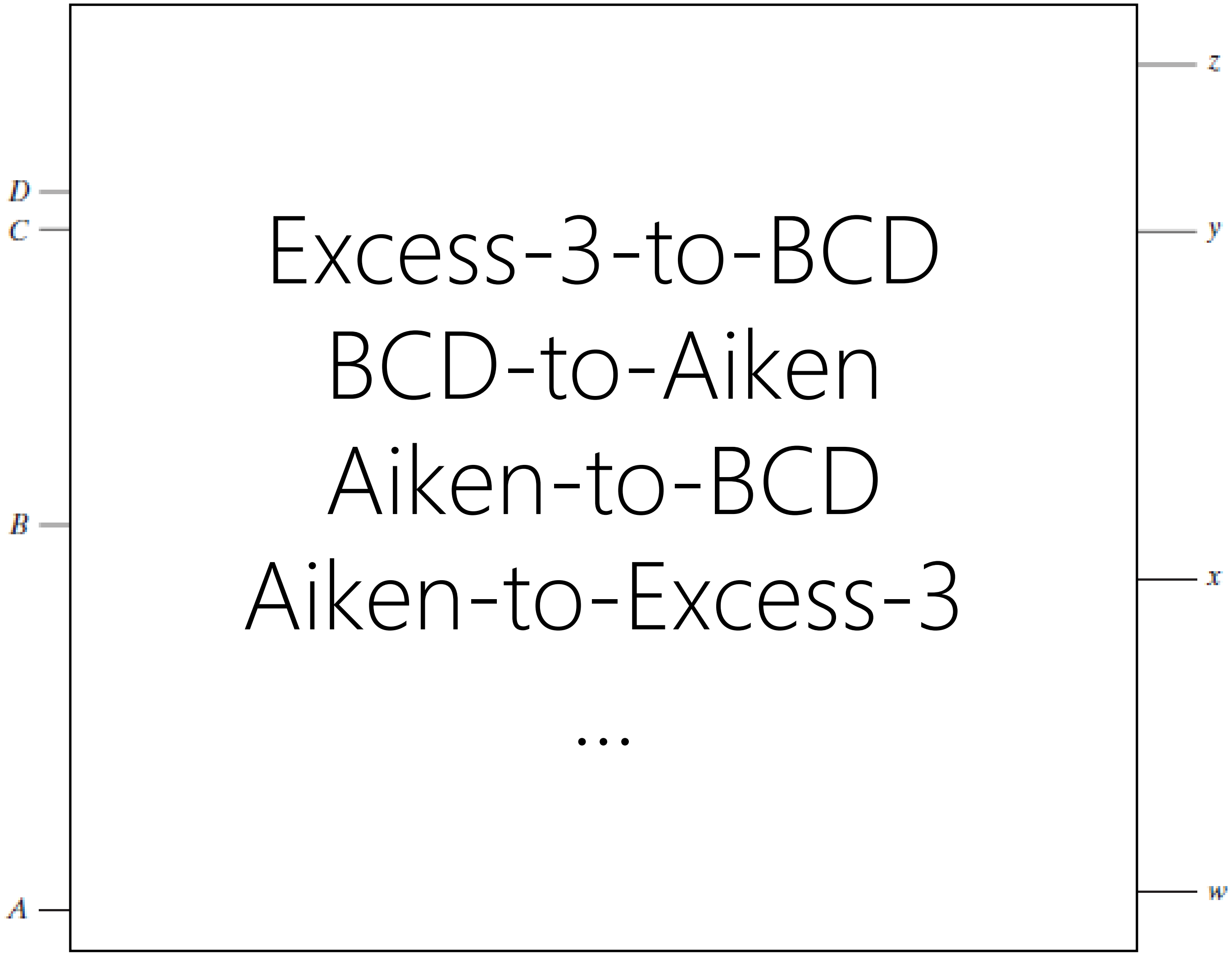
$$X(A,B,C,D) = \sum(1,2,3,4,9) + d(10,11,12,13,14,15)$$
$$= \prod(0,5,6,7,8) + D(10,11,12,13,14,15)$$
$$= ((BD)+(BC)+(B'C'D'))'$$ Here the "don't care conditions" helped ☺
$$= (B'+D')(B'+C')(B+C+D)$$

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

$$Y(A,B,C,D) = \sum(0,3,4,7,8) + d(10,11,12,13,14,15)$$
$$= ?\ \prod(\ ?\ ) + D$$

$$Z(A,B,C,D) = \sum(0,2,4,6,8) + d(10,11,12,13,14,15)$$
$$= \prod ?\ (\ ?\ ) + D$$

Your Turn!


D
C
B
A
Excess-3-to-BCD
BCD-to-Aiken
Aiken-to-BCD
Aiken-to-Excess-3
...
z
y
x
w

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?

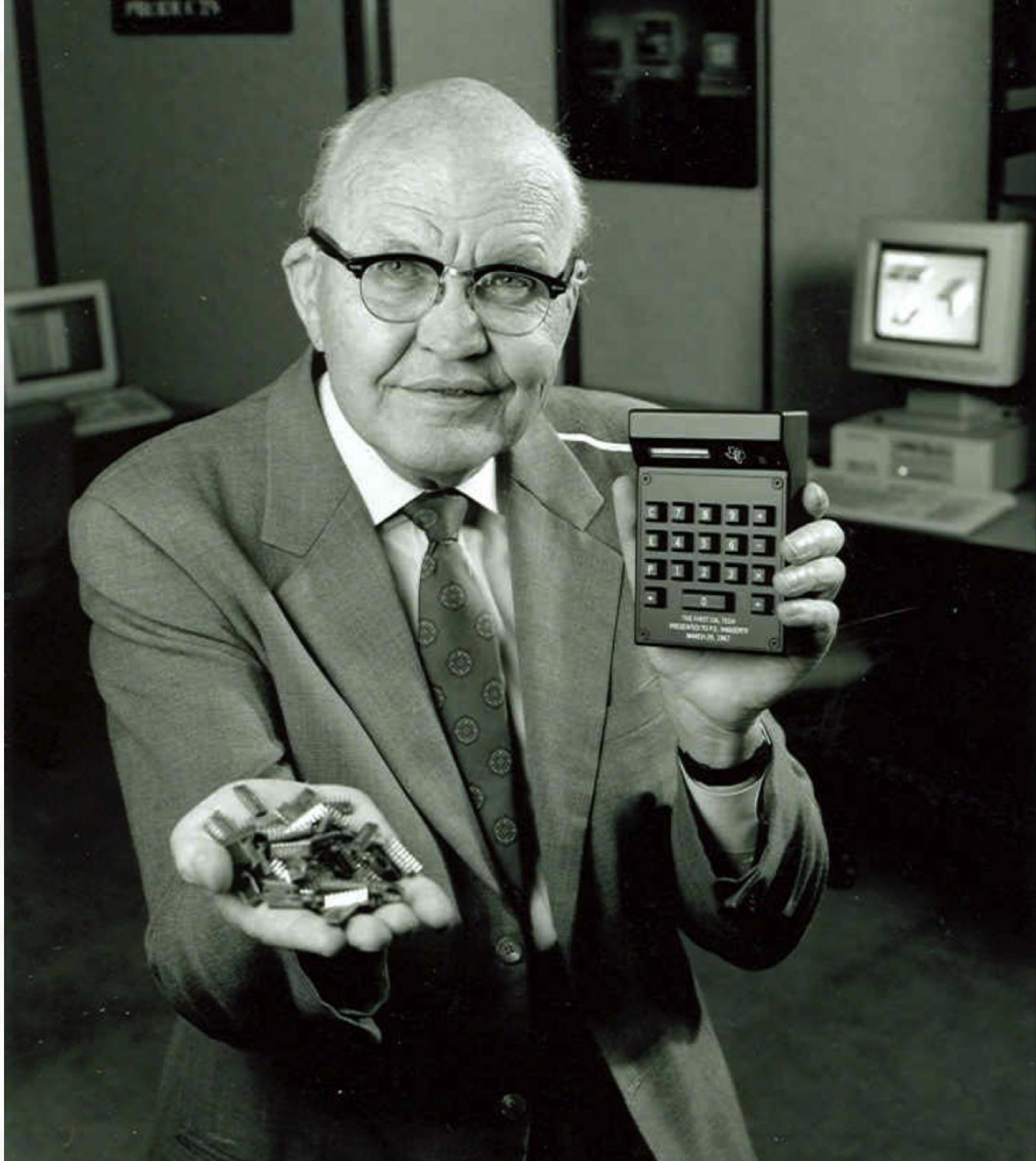
Bowmar was the first American company that made and sold their own line of portable electronic machines.

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



Combinational Logic

Binary Code Arithmetic

Combinational Logic

BCD Adder

Book: Page 144-146