Number Representation and Arithmetic Circuits

Chapter 5

Chapter Objectives

- Representation of numbers in computers
- Circuits used to perform arithmetic operations
- Performance issues in large circuit

Contents

NUMBER REPRESENTATIONS IN DIGITAL SYSTEMS

Number Representation

- Unsigned integers

$$B = b_{n-1}b_{n-2}\cdots b_1 b_0$$

$$V(B) = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \cdots + b_1 \times 2^1 + b_0 \times 2^0$$

$$= \sum_{i=0}^{n-1} b_i \times 2^i$$

- Octal and Hexadecimal Representation

$$K = k_{n-1}k_{n-2}\cdots k_1 k_0$$

$$V(K) = \sum_{i=0}^{n-1} k_i \times r^i$$

Numbers in different systems

Decimal	Binary	Octal	Hexadecimal
00	000000	00	00
01	000001	01	01
02	000010	02	02
03	000011	03	03
04	000100	04	04
05	000101	05	05
06	000110	06	06
07	000111	07	07
08	001000	10	08
09	001001	11	09
10	001010	12	$0A$
11	001011	13	$0B$
12	001100	14	$0C$
13	001101	15	$0D$
14	001110	16	$0E$
15	001111	17	$0F$
16	010000	20	10
17	010001	21	11
18	010010	22	12

Figure 5.1. Numbers in different systems.

Conversion from decimal to binary

Convert $(857)_{10}$

	Remainder	
$857 \div 2 = 428$	1	LSB (Least Significant Bit)
$428 \div 2 = 214$	0	
$214 \div 2 = 107$	0	
$107 \div 2 = 53$	1	
$53 \div 2 = 26$	1	
$26 \div 2 = 13$	0	
$13 \div 2 = 6$	1	
$6 \div 2 = 3$	0	
$3 \div 2 = 1$	1	
$1 \div 2 = 0$	1	MSB (Most Significant Bit)

Result is $(1101011001)_2$

Octal and Hexadecimal Numbers

Conversion binary number to octal and hexadecimal number

$$\left(\underbrace{10}_{2}\ \underbrace{110}_{6}\ \underbrace{001}_{1}\ \underbrace{101}_{5}\ \underbrace{011}_{3}.\underbrace{111}_{7}\ \underbrace{100}_{4}\ \underbrace{000}_{0}\ \underbrace{110}_{6}\right)_2 = (26153.7406)_8$$

$$\left(\underbrace{10}_{2}\ \underbrace{1100}_{C}\ \underbrace{0110}_{6}\ \underbrace{1011}_{B}.\underbrace{1111}_{F}\ \underbrace{0000}_{0}\ \underbrace{0110}_{6}\right)_2 = (2C6B.F06)_{16}$$

ADDITION OF UNSIGNED NUMBERS

Addition of Unsigned Numbers

$$\begin{array}{r} x \\ +\ y \\ \hline c\ \ s \end{array} \qquad \begin{array}{r} 0 \\ +\ 0 \\ \hline 0\ \ 0 \end{array} \qquad \begin{array}{r} 0 \\ +\ 1 \\ \hline 0\ \ 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 0 \\ \hline 0\ \ 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 1 \\ \hline 1\ \ 0 \end{array}$$

(a) The four possible cases

x	y	carry c	Sum s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(b) Truth table

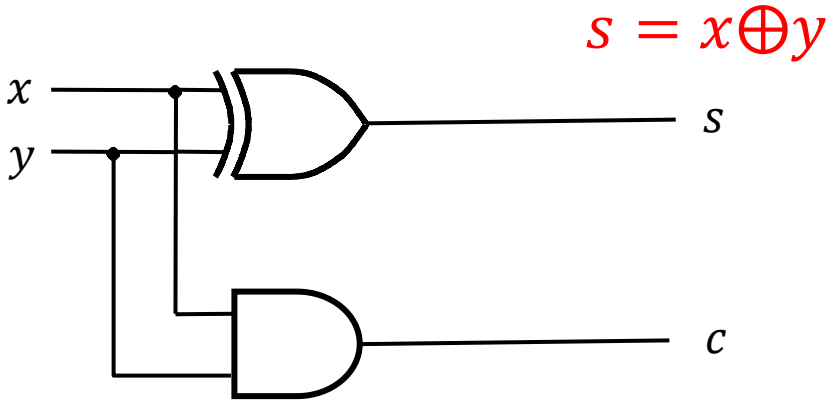
$$c = xy$$

$$s = x'y + xy'$$
$$= x \oplus y$$

Mod 2 addition

Figure 5.2. Half-adder.

Addition of Unsigned Numbers



(c) Circuit

(d) Graphical symbol

Figure 5.2. Half-adder.

An example of addition

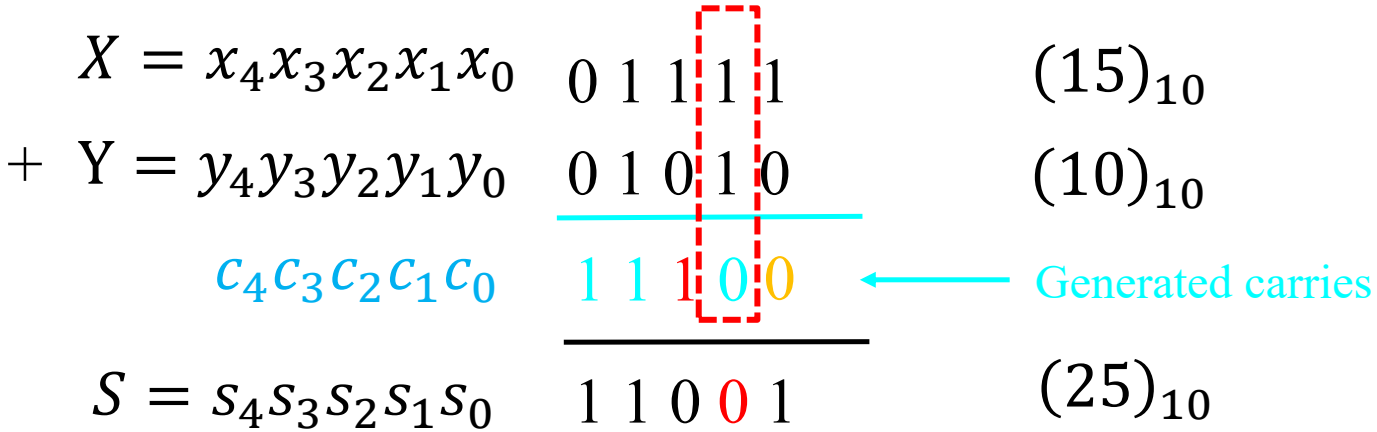
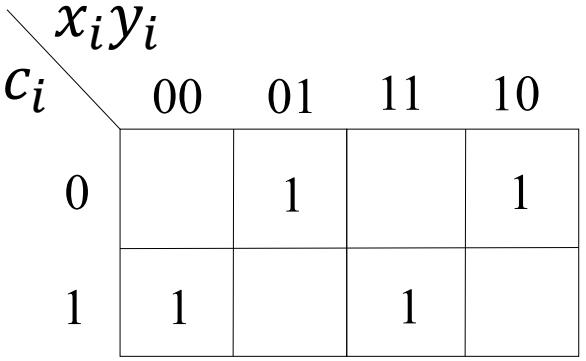


Figure 5.3. An example of addition.

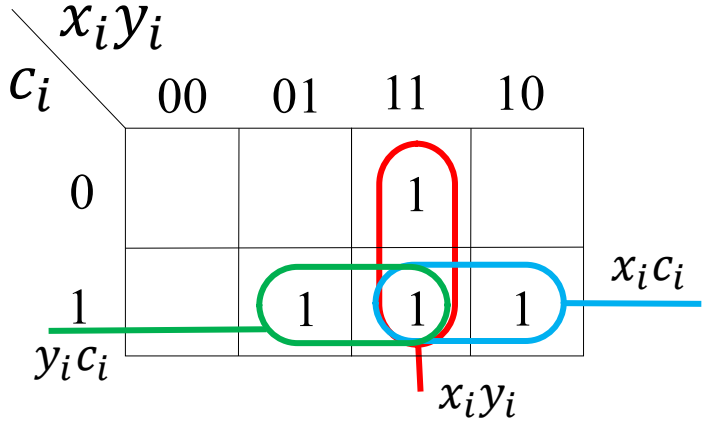
Full Adder

c_i	x_i	y_i	c_{i+1}	s_i
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

(a) Truth table



$$s_i = x_i \oplus y_i \oplus c_i$$



$$c_{i+1} = x_iy_i + x_ic_i + y_ic_i$$

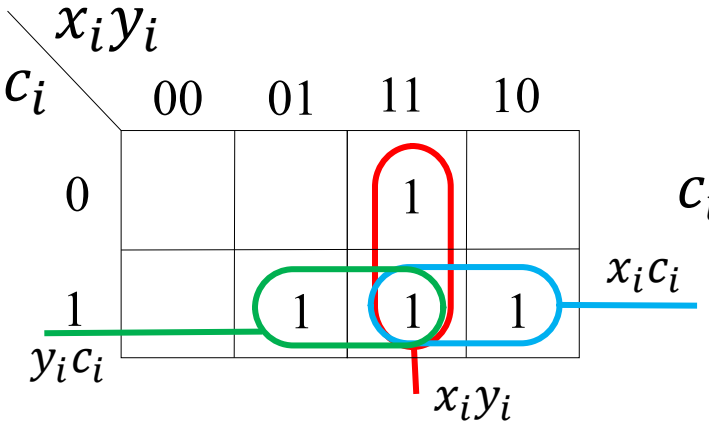
(b) Karnaugh maps

Figure 5.4. Full-adder.

Full Adder

c_i \ x_iy_i	00	01	11	10
0		1		1
1	1		1	

$$s_i = x_i \oplus y_i \oplus c_i$$



$$c_{i+1} = x_iy_i + x_ic_i + y_ic_i$$

$$s_i = c_i'x_i'y_i + c_i'x_iy_i' + c_ix_i'y_i' + c_ix_iy_i$$

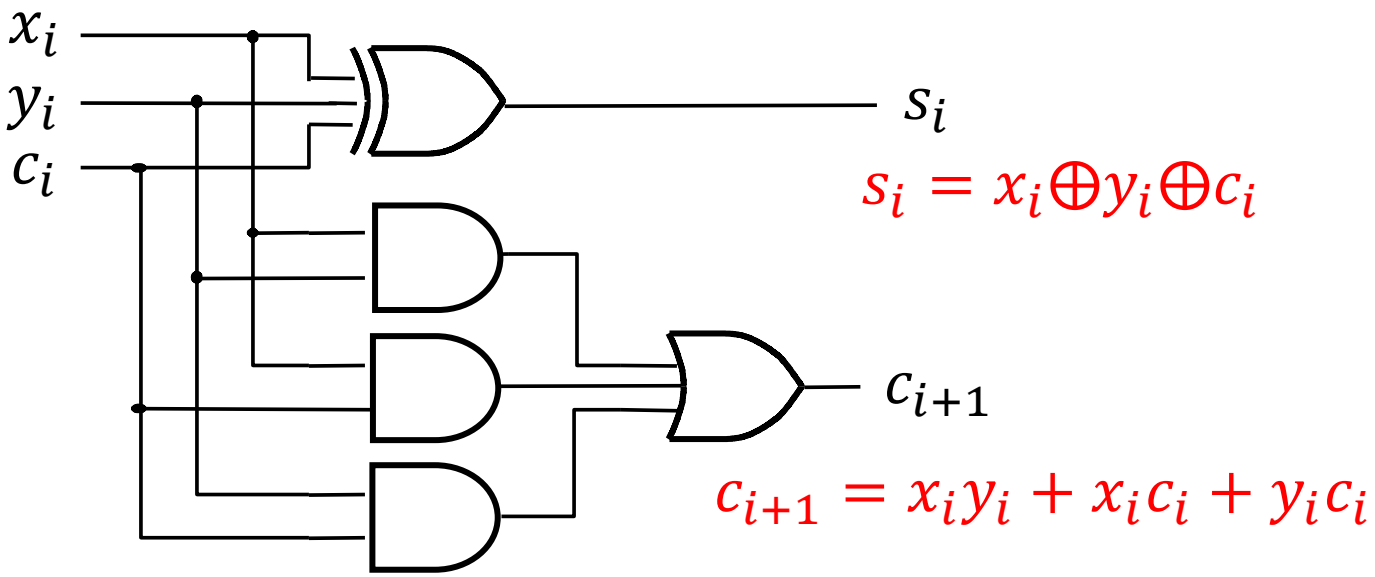
$x_i \oplus y_i = x_i'y_i + x_iy_i'$ X-OR

$(x_i \oplus y_i)' = x_i'y_i' + x_iy_i$ X-NOR

$$\begin{aligned} s_i &= c_i'(x_i'y_i + x_iy_i') + c_i(x_i'y_i' + x_iy_i) \\ &= c_i'(x_i \oplus y_i) + c_i(x_i \oplus y_i)' \\ &= x_i \oplus y_i \oplus c_i \end{aligned}$$

$$c_{i+1} = c_i'x_iy_i + c_ix_i'y_i + c_ix_iy_i' + c_ix_iy_i$$

Full Adder



(c) Circuit

Figure 5.4. Full-adder.

Decomposed Full-Adder

HA

$s = x \oplus y$

$c = xy$

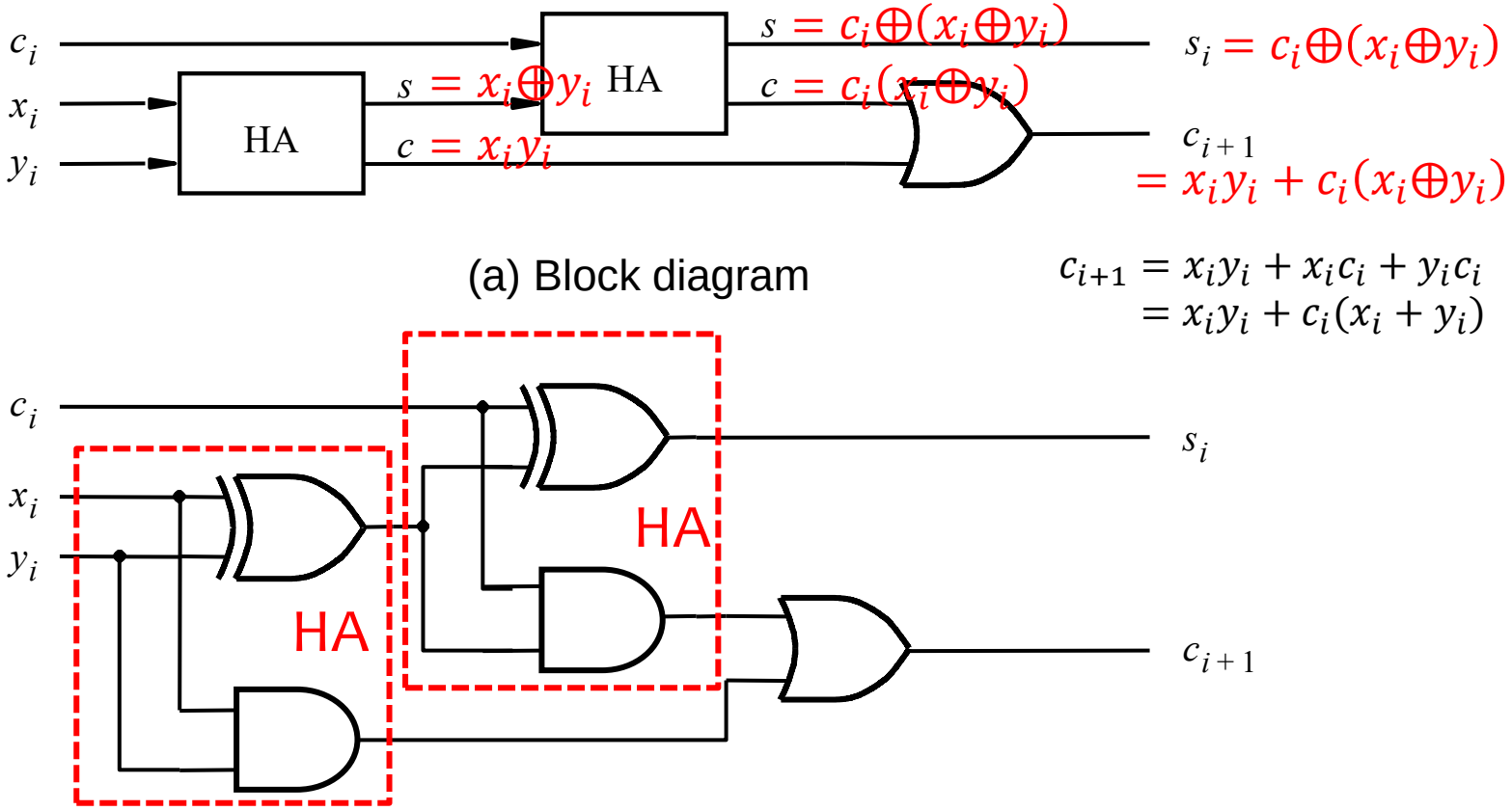


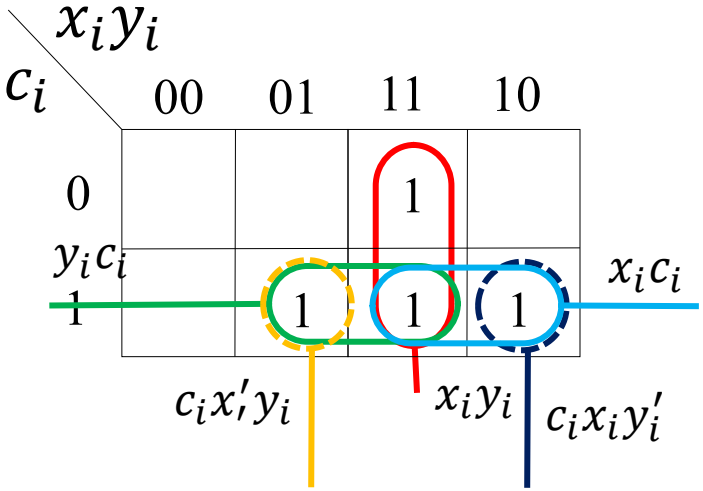
Figure 5.5 A decomposed implementation of the full adder circuit.

Implementation of c_{i+1}

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$
$$= x_i y_i + c_i(x_i + y_i)$$

$$c_{i+1} = x_i y_i + c_i(x_i \oplus y_i)$$
$$= x_i y_i + c_i x_i' y_i + c_i x_i y_i'$$
$$= x_i y_i + x_i c_i + y_i c_i$$

$$x_i \oplus y_i = x_i' y_i + x_i y_i'$$



An example of addition

	$X = x_4x_3x_2x_1x_0$	0 1 1 1 1	$(15)_{10}$
+	$Y = y_4y_3y_2y_1y_0$	0 1 0 1 0	$(10)_{10}$
	$c_5\ c_4\ c_3\ c_2\ c_1$	0 1 1 1 0	
		(4) (3) (2) (1)	
	$S = s_4\ s_3\ s_2\ s_1\ s_0$	1 1 0 0 1	$(25)_{10}$

Figure 5.3. An example of addition.

Ripple Carry Adder

The delay depends on the size of numbers

FA: full adder

$$Total\ delay \approx n\Delta t, \Delta t\ is\ one\ bit\ delay$$

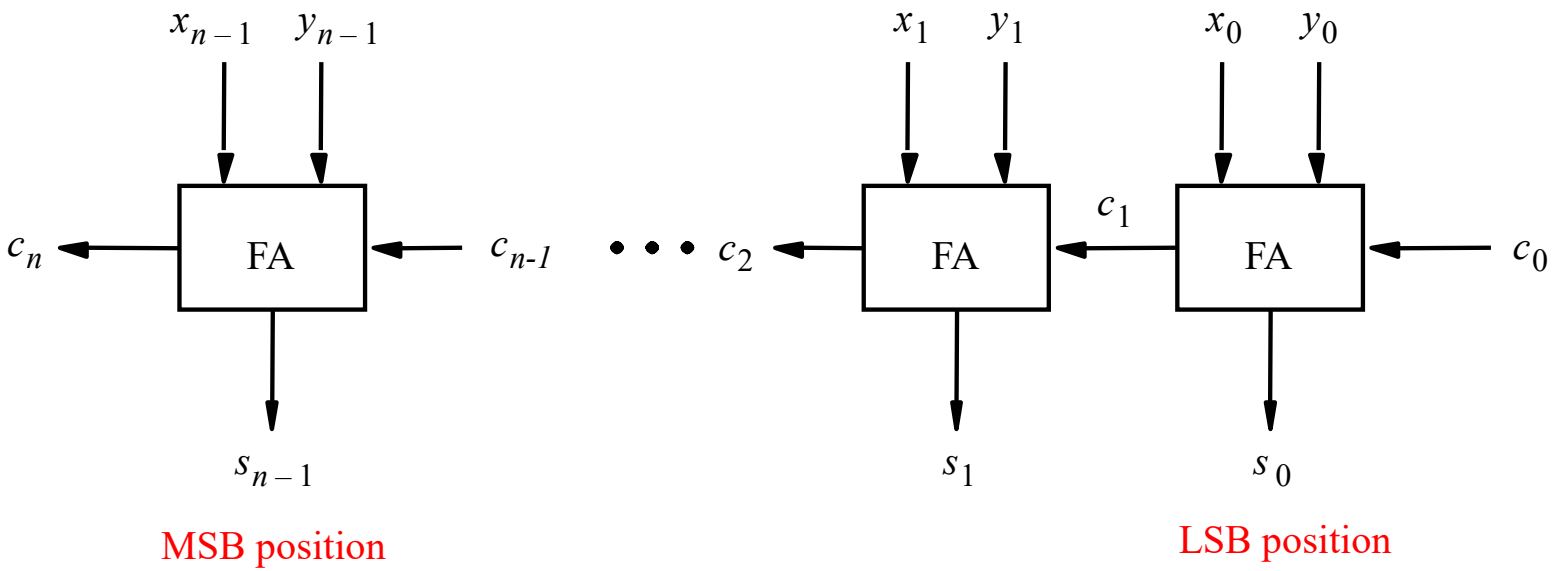
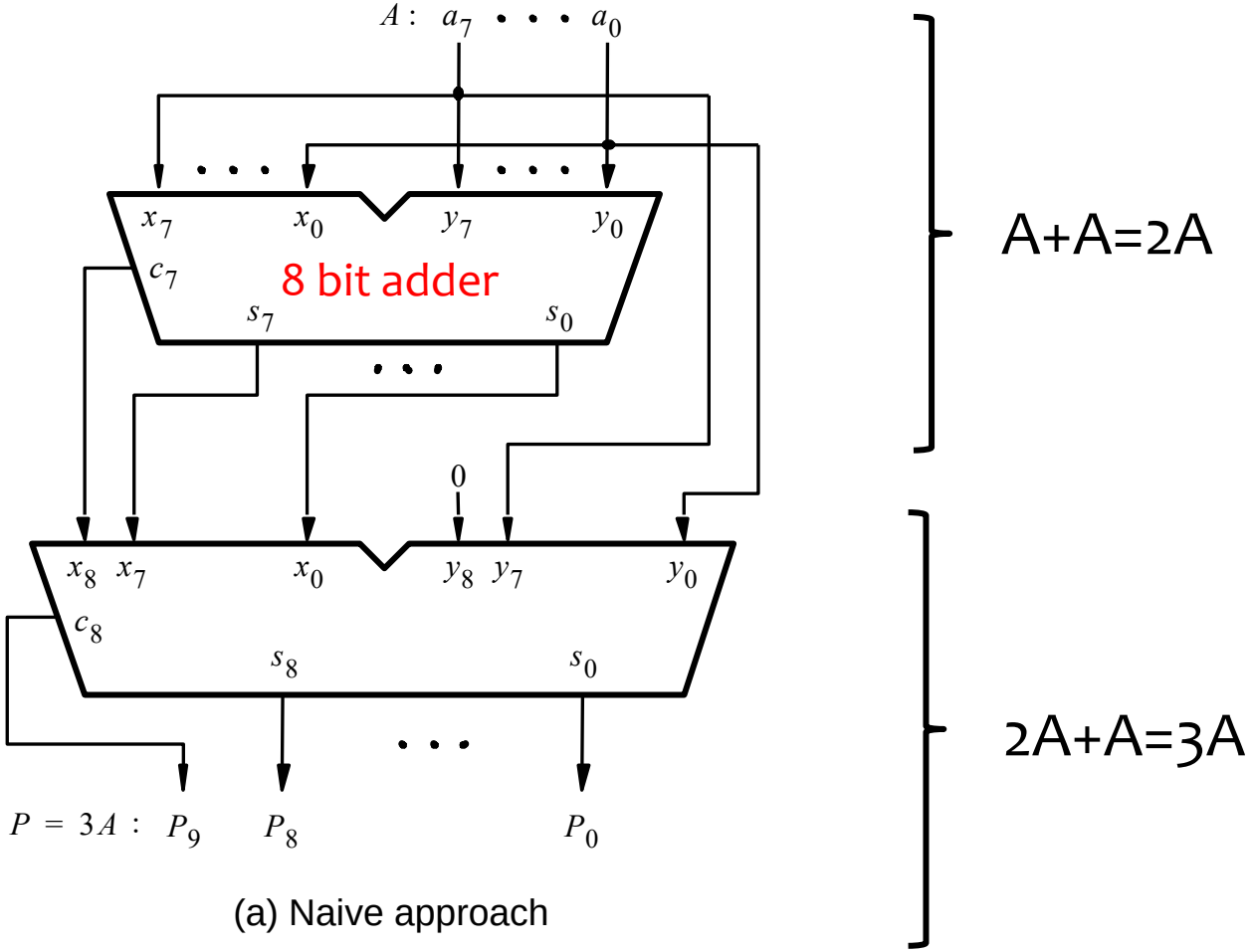


Figure 5.6. An *n*-bit ripple-carry adder.

Design Example

The Logic to calculate 3A using 8 bit adder

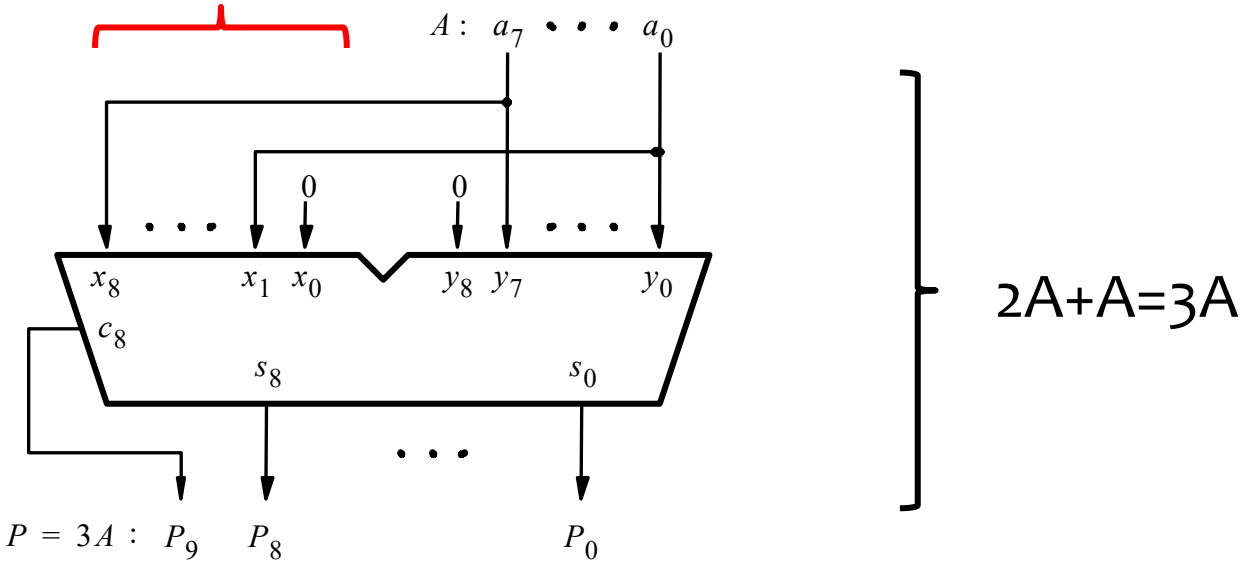


(a) Naive approach

Figure 5.7. Circuit that multiplies an eight-bit unsigned number by 3.

Design Example

Shift by 1 to the Left = 2A

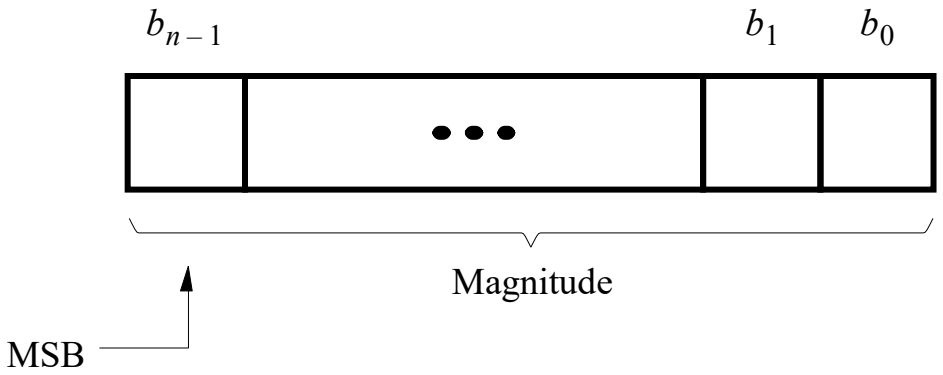


(b) Efficient design

Figure 5.7. Circuit that multiplies an eight-bit unsigned number by 3.

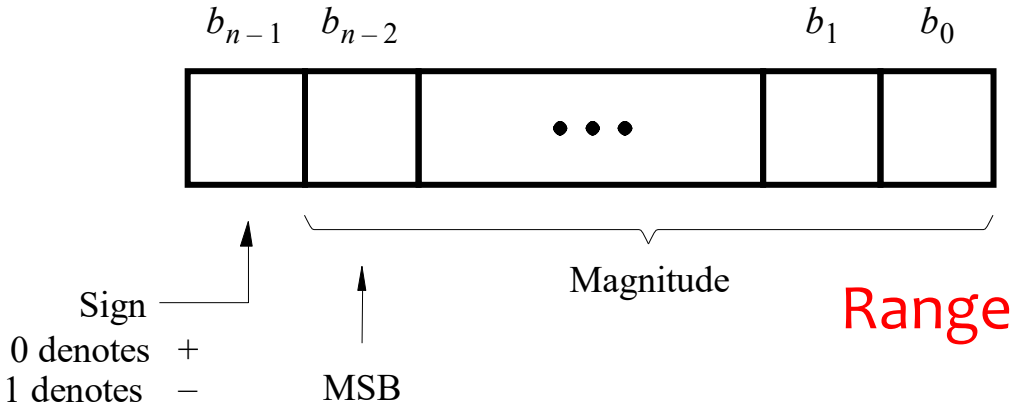
SIGNED NUMBERS

Signed Numbers



Range: $0 \sim 2^n - 1$

(a) Unsigned number



Range: $-2^{n-1} \sim 2^{n-1} - 1$

(b) Signed number

Figure 5.8. Formats for representation of integers.

Signed Number

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

Sign and magnitude
: $-(2^{n-1}-1)$ ~$2^{n-1}-1$

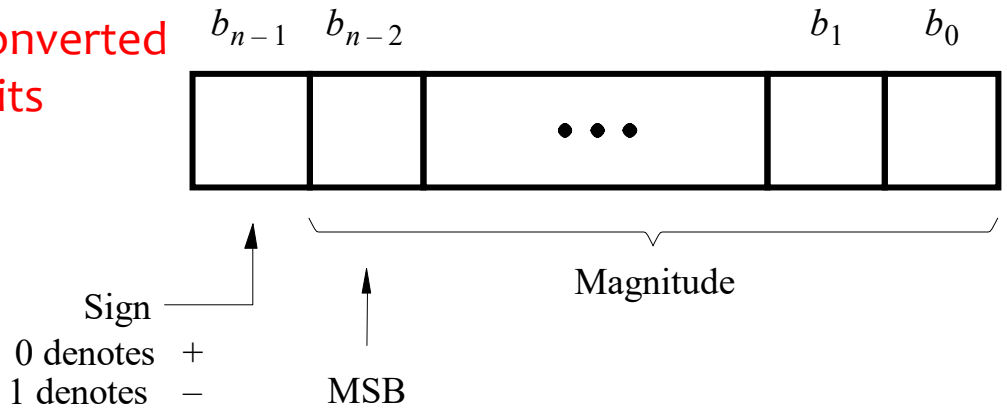
2's complement
: -2^{n-1} ~$2^{n-1}-1$

1's complement
: $+0 = 0000$
$-0 = 1111$

Table 5.1. Interpretation of four-bit signed integers.

Sign and Magnitude

N : number to be converted
n : the number of bits



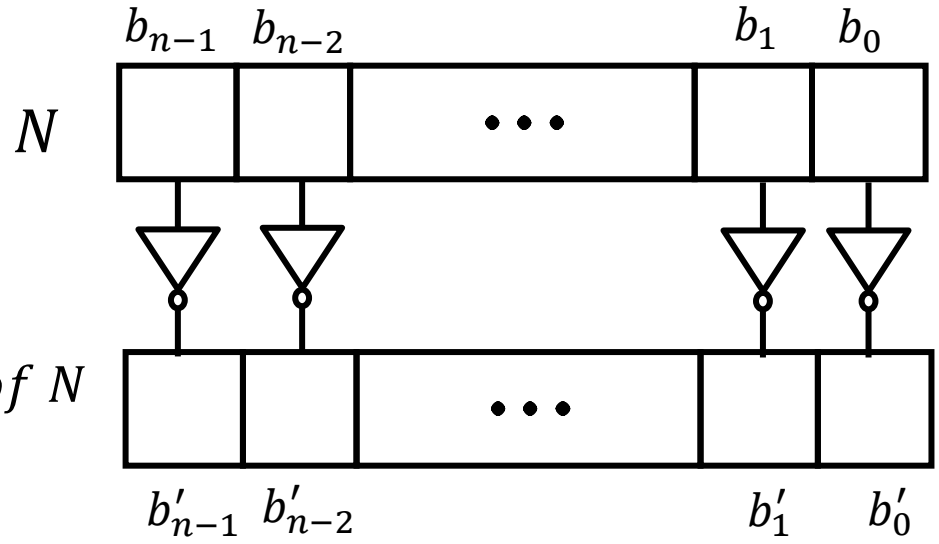
When n= 4

Sign bit

$(0\ 1\ 0\ 1)_2$ $(5)_{10}$

$(1\ 1\ 0\ 1)_2$ $(-5)_{10}$

1's Complement



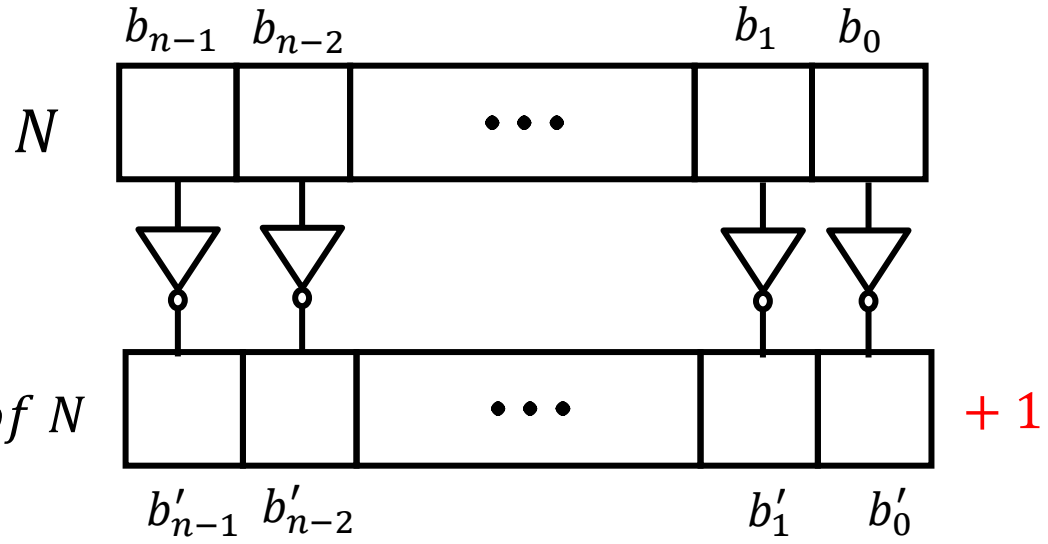
$1's\ complement\ of\ N = (2^n - 1) - N$

When n= 4

$$(2^4 - 1) = (1\ \ 1\ \ 1\ \ 1)_2 \qquad (15)_{10}$$
$$-\ \ N = (0\ \ 1\ \ 0\ \ 1)_2 \qquad (5)_{10}$$

$$(1\ \ 0\ \ 1\ \ 0)_2 \qquad (-5)_{10}$$

2's Complement



$$2's\ complement\ of\ N = 2^n - N = \left((2^n - 1) - N\right) + 1$$
$$= 1's\ complement\ of\ N + 1$$

When n= 4

$$\begin{array}{rll} (2^4 - 1) = & (1\ \ 1\ \ 1\ \ 1)_2 & (15)_{10} \\ - \quad N = & (0\ \ 1\ \ 0\ \ 1)_2 & (5)_{10} \\ + & (0\ \ 0\ \ 0\ \ 1)_2 & \\ \hline & (1\ \ 0\ \ 1\ \ 1)_2 & (-5)_{10} \end{array}$$

1's complement addition

$r = 2$
$n = 4$

r : Base
N : number to be converted
n : the number of bits

$-M = (2^4 - 1) - M$
$-N = (2^4 - 1) - N$

$M > N$
$M + N \leq (2^4 - 1)$

$M + N$ positive

$$\begin{array}{rr} (+5) & 0\ 1\ 0\ 1 \\ +(+2) & +\ 0\ 0\ 1\ 0 \\ \hline (+7) & 0\ 1\ 1\ 1 \end{array}$$

$-M + N = -(M - N)$ negative

$$\begin{array}{rr} (-5) & 1\ 0\ 1\ 0 \\ +(+2) & +\ 0\ 0\ 1\ 0 \\ \hline (-3) & 1\ 1\ 0\ 0 \end{array}$$

$-M + N$
$= (2^4 - 1) - M + N$
$= (2^4 - 1) - (M - N)$

No carry !

Figure 5.9. Examples of 1's complement addition.

1's complement addition

$M - N$ positive

$$
\begin{array}{rr}
(+5) & 0\ 1\ 0\ 1 \\
+(-2) & +\ 1\ 1\ 0\ 1 \\
\hline
(+3) & 1\ 0\ 0\ 1\ 0 \\
 & \quad\ \ \rightarrow 1 \\
\hline
 & 0\ 0\ 1\ 1
\end{array}
$$

Carry generated!

$$M - N = M + (2^4 - 1) - N = M - N + (2^4 - 1)$$

$-M - N = -(M + N)$ negative

$$
\begin{array}{rr}
(-5) & 1\ 0\ 1\ 0 \\
+(-2) & +\ 1\ 1\ 0\ 1 \\
\hline
(-7) & 1\ 0\ 1\ 1\ 1 \\
 & \quad\ \ \rightarrow 1 \\
\hline
 & 1\ 0\ 0\ 0
\end{array}
$$

Carry generated!

$$-M - N = (2^4 - 1) - M + (2^4 - 1) - N = (2^4 - 1) - (M + N) + (2^4 - 1)$$

Figure 5.9. Examples of 1's complement addition.

2's complement addition

$M + N$ positive $-M + N = -(M - N)$ negative

(+5)	0 1 0 1	(−5)	1 0 1 1
+(+2)	+ 0 0 1 0	+(+2)	+ 0 0 1 0
(+7)	0 1 1 1	(−3)	1 1 0 1

No carry !

$$-M + N$$
$$= 2^4 - M + N$$
$$= 2^4 - (M - N)$$

Figure 5.10. Examples of 2's complement addition.

2's complement addition

$M - N$ positive

$$\begin{array}{rr} (+5) & 0\ 1\ 0\ 1 \\ +(-2) & +\ 1\ 1\ 1\ 0 \\ \hline (+3) & 1\ 0\ 0\ 1\ 0 \end{array}$$

Carry generated!

ignore

$$M - N = M + 2^4 - N = M - N + 2^4$$

$-M - N = -(M + N)$ negative

$$\begin{array}{rr} (-5) & 1\ 0\ 1\ 1 \\ +(-2) & +\ 1\ 1\ 1\ 0 \\ \hline (-7) & 1\ 1\ 0\ 0\ 1 \end{array}$$

Carry generated!

ignore

$$-M - N = 2^4 - M + 2^4 - N = 2^4 - (M + N) + 2^4$$

Figure 5.10. Examples of 2's complement addition.

2's complement subtraction

(+5)	0 1 0 1	⇒	0 1 0 1	M
−(−2)	− 1 1 1 0		+ 0 0 1 0	$+ \quad N$
(+7)			0 1 1 1	$= M + N$

(−5)	1 0 1 1	⇒	1 0 1 1	$2^4 - M$
−(−2)	− 1 1 1 0		+ 0 0 1 0	$+ \quad N$
(−3)			1 1 0 1	$= 2^4 - (M - N)$

Figure 5.11. Examples of 2's complement subtraction.

2's complement subtraction

$$\begin{array}{rr} (+5) & 0\ 1\ 0\ 1 \\ -(+2) & -\ 0\ 0\ 1\ 0 \\ \hline (+3) & \end{array} \Longrightarrow \begin{array}{r} 0\ 1\ 0\ 1 \\ +\ 1\ 1\ 1\ 0 \\ \hline 1\ 0\ 0\ 1\ 1 \end{array} \quad \begin{array}{r} M \\ +\ (2^4 - N) \\ \hline = M - N + 2^4 \end{array}$$

ignore

$$\begin{array}{rr} (-5) & 1\ 0\ 1\ 1 \\ -(+2) & -\ 0\ 0\ 1\ 0 \\ \hline (-7) & \end{array} \Longrightarrow \begin{array}{r} 1\ 0\ 1\ 1 \\ +\ 1\ 1\ 1\ 0 \\ \hline 1\ 1\ 0\ 0\ 1 \end{array} \quad \begin{array}{r} 2^4 - M \\ +\ (2^4 - N) \\ \hline = 2^4 - (M + N) + 2^4 \end{array}$$

ignore

Figure 5.11. Examples of 2's complement subtraction.

4 bit 2's complement
: −8~ + 7

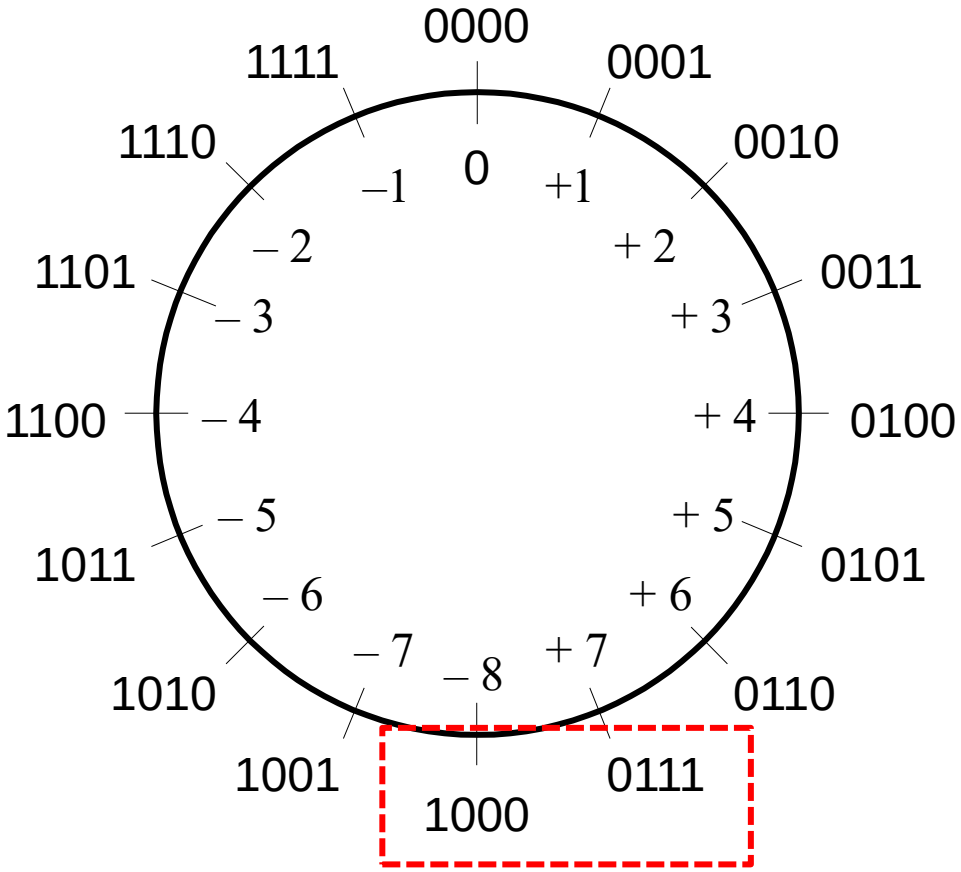
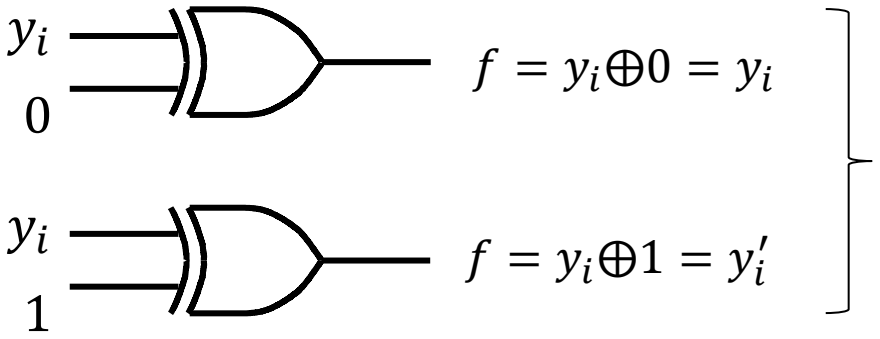


Figure 5.12. Graphical interpretation of four-bit 2's complement numbers.

Adder/Subtractor Unit

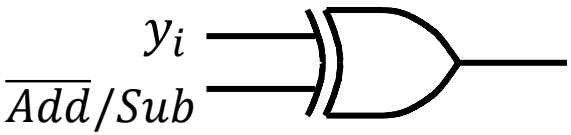
2's complement = 1's complement + 1

1's complement : bit inversion (NOT gate)



0: addition

1: subtraction



$$f = y_i \oplus \overline{Add}/Sub = \begin{cases} y_i & if \overline{Add}/Sub = 0 \\ y_i' & if \overline{Add}/Sub = 1 \end{cases}$$

Adder/Subtractor Unit

$n\ bit + n\ bit = n + 1\ bit$

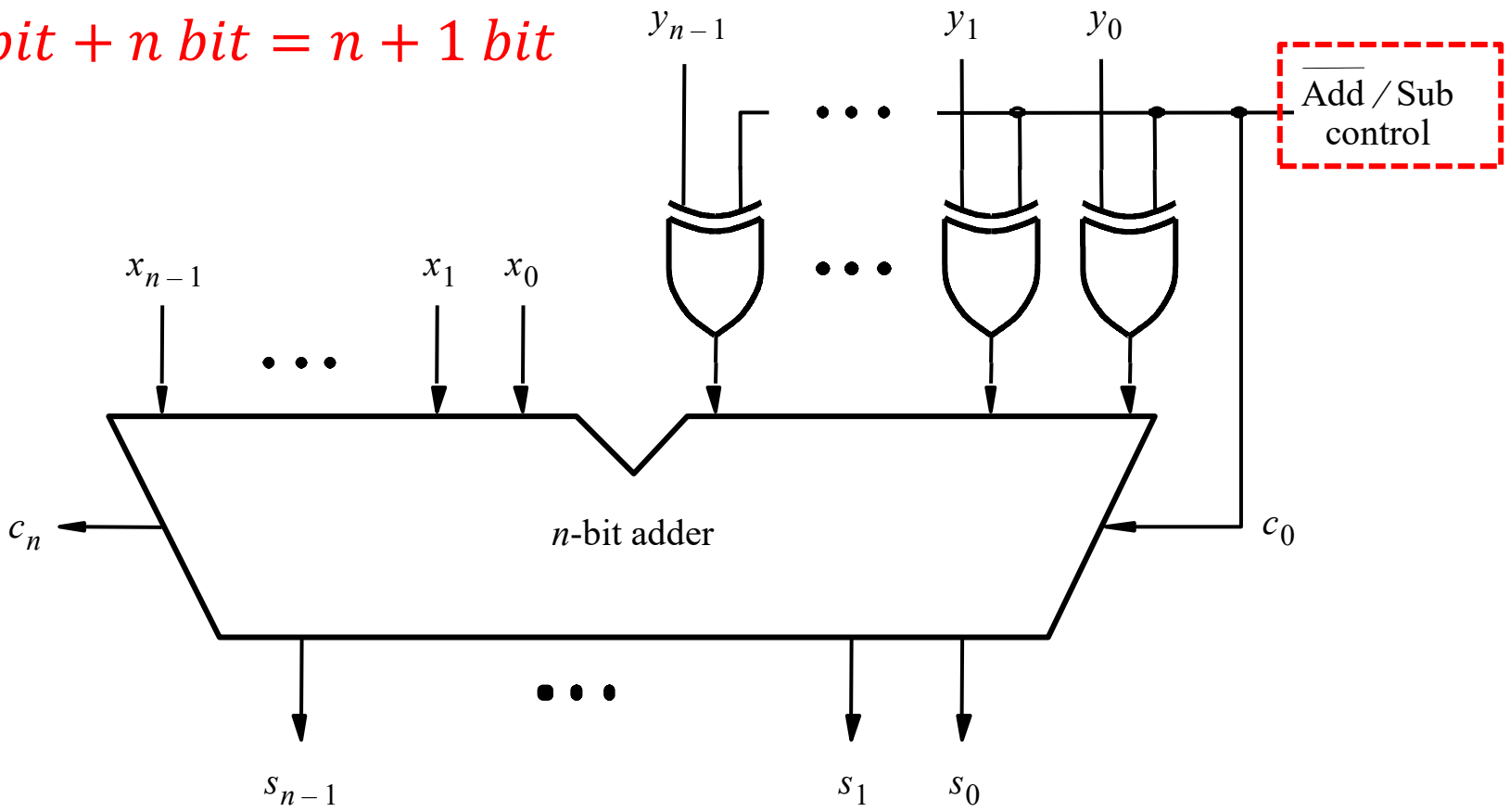


Figure 5.13. Adder/subtractor unit.

Arithmetic Overflow

Range of 4 bit 2'c complement binary number

: $-2^3 \sim 2^3 - 1 = -8 \sim +7$

Overflow = $c_{n-1} \oplus c_n$

Overflow : Out of range

(+7)	0 1 1 1	(–7)	1 0 0 1
+ (+2)	+ 0 0 1 0	+ (+2)	+ 0 0 1 0
(+9)	1 0 0 1	(–5)	1 0 1 1

$c_4 = 0$, $c_3 = 1$ Overflow !!!

$c_{n-1} \oplus c_n = 1 \oplus 0 = 1$

$c_4 = 0$, $c_3 = 0$

$c_{n-1} \oplus c_n = 0 \oplus 0 = 0$

(+7)	0 1 1 1	(–7)	1 0 0 1
+ (–2)	+ 1 1 1 0	+ (–2)	+ 1 1 1 0
(+5)	1 0 1 0 1	(–9)	1 0 1 1 1

$c_4 = 1$, $c_3 = 1$

$c_{n-1} \oplus c_n = 1 \oplus 1 = 0$

$c_4 = 1$, $c_3 = 0$ Overflow !!!

$c_{n-1} \oplus c_n = 0 \oplus 1 = 1$

Figure 5.14. Examples of determination of overflow.

Performance Issues

- When buying a digital system, the buyer pays particular attention to the performance and the cost.
- Superior performance comes at a higher cost.
- In n-bit ripple carry adder, the delay is approximately $n\Delta t$, where Δt is a one bit delay.
- To speed up the computer performance, it is important to find faster circuits to perform addition.

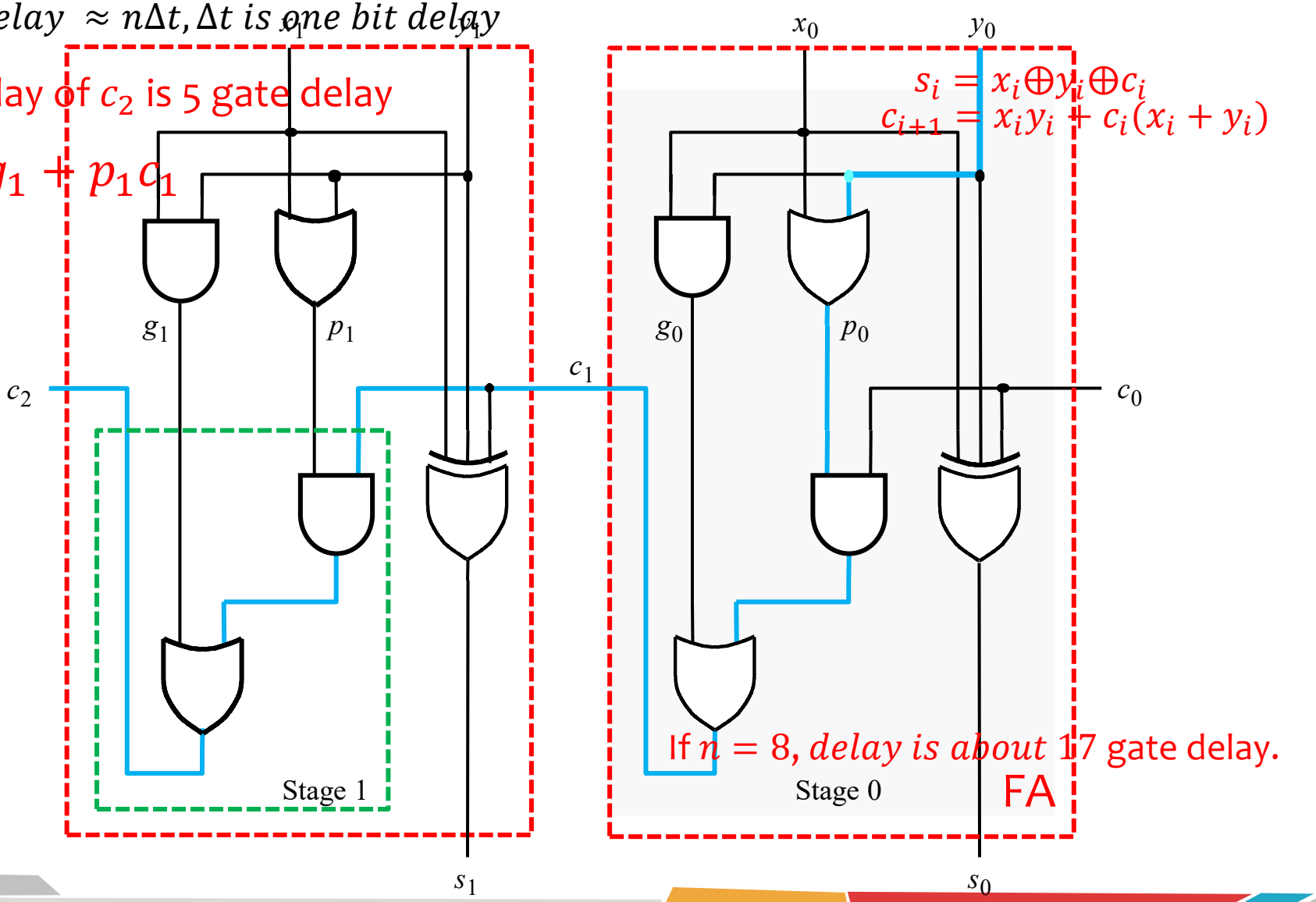
FAST ADDER

Ripple-Carry Adder

$Total\ delay \approx n\Delta t, \Delta t\ is\ one\ bit\ delay$

The delay of c_2 is 5 gate delay

$c_2 = g_1 + p_1 c_1$



If $n = 8$, *delay is about* 17 gate delay.

Figure 5.15. A ripple-carry adder based on expression 5.3.

Carry-Lookahead Adder

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$c_{i+1} = x_i y_i + c_i(x_i + y_i)$$

$$c_{i+1} = g_i + p_i c_i$$

Ripple Carry Adder

$g_i = x_i y_i$: generate function — 1 x 2-input AND

$p_i = x_i + y_i$: propagate function — 1 x 2-input OR

$$c_{i+1} = g_i + p_i(g_{i-1} + p_{i-1} c_{i-1})$$

$$c_{i+1} = g_i + p_i g_{i-1} + p_i p_{i-1} c_{i-1}$$

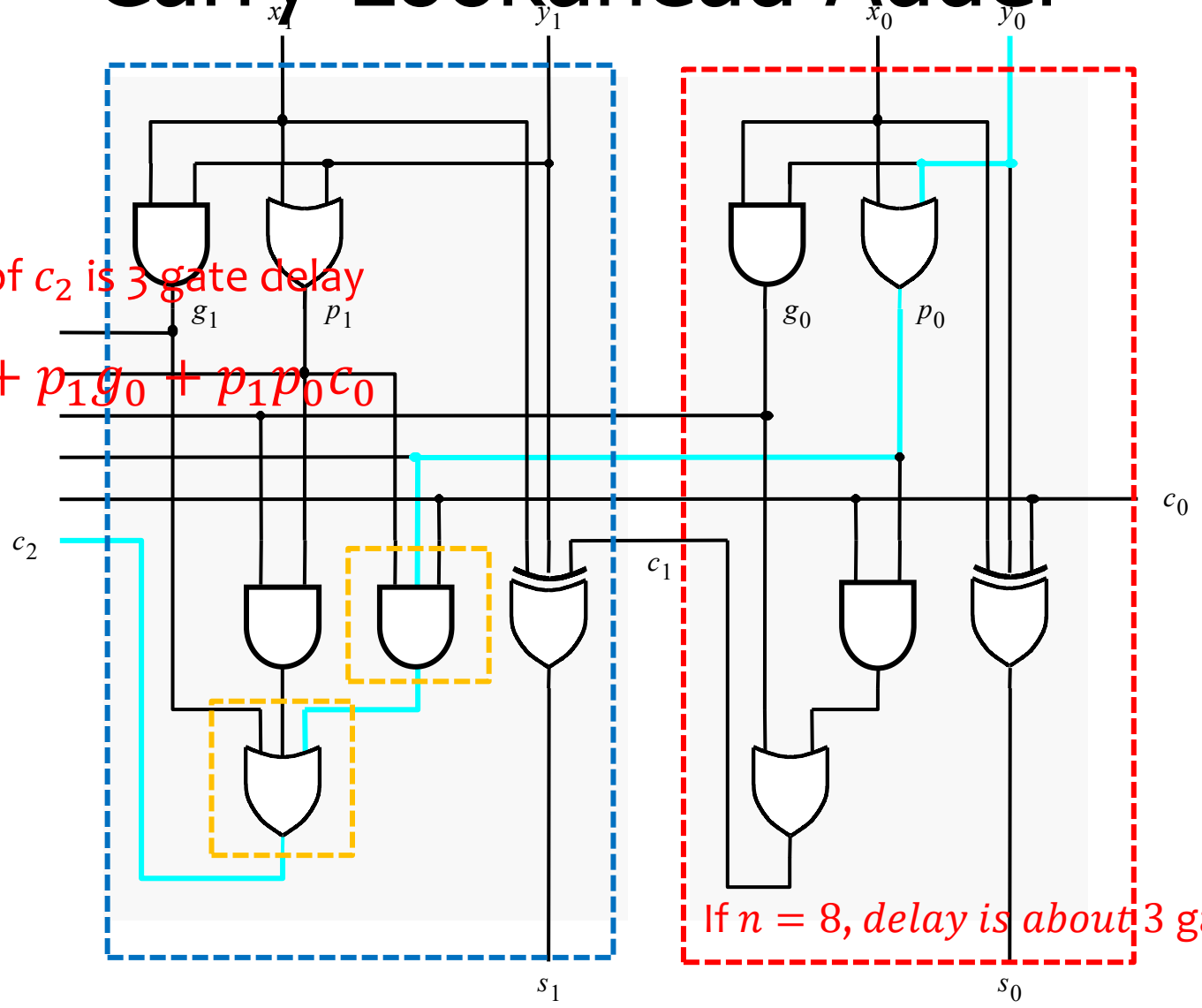
$$\vdots$$

$$c_{i+1} = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + \cdots + p_i p_{i-1} \cdots p_2 p_1 g_0 + p_i p_{i-1} \cdots p_1 p_0 c_0$$

Carry-Lookahead Adder

The delay of c_2 is 3 gate delay

$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$



If $n = 8$, *delay is about* 3 gate delay.

Figure 5.16. The first two stages of a carry-lookahead adder.

Carry-Lookahead Adder

$$c_{i+1} = g_i + p_i c_i$$

$$c_1 = g_0 + p_0 c_0$$

$$c_2 = g_1 + p_1 c_1$$ Ripple carry

$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$ Carry Lookahead

$$c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$$

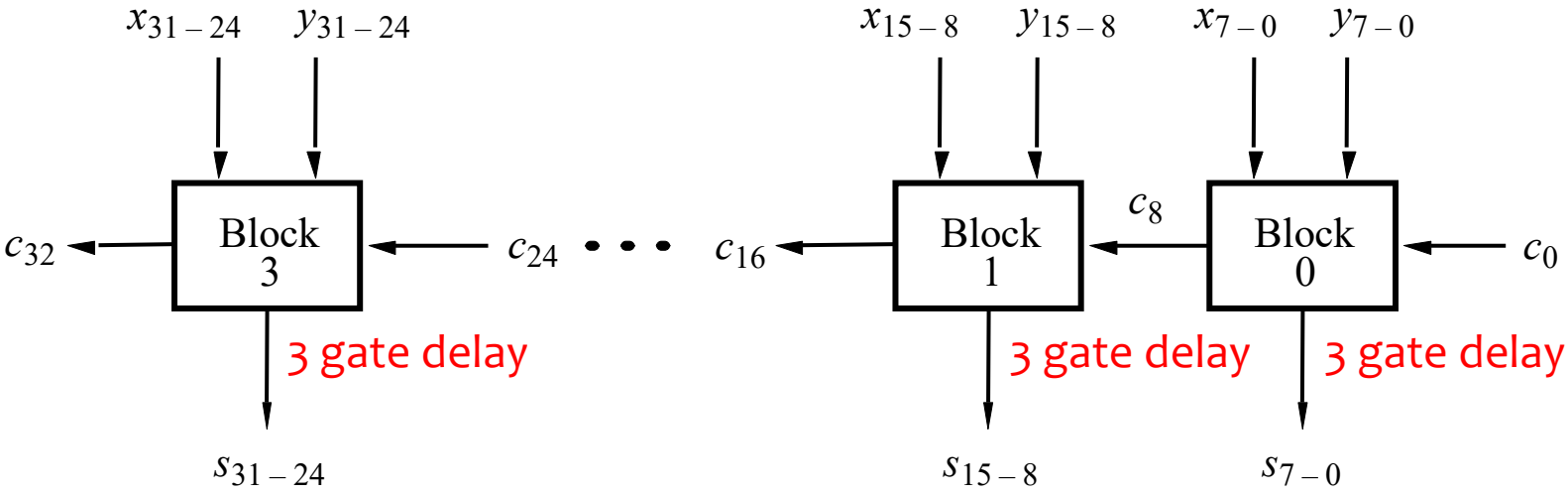
$$c_{i+1} = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + \cdots + p_i p_{i-1} \cdots p_2 p_1 g_0 + p_i p_{i-1} \cdots p_1 p_0 c_0$$

- The complexity of an n-bit-carry-lookahead adder increases rapidly as n becomes larger.
- To reduce the complexity, a hierarchical approach can be used.

hierarchical carry-lookahead adder

Carry-lookahead inside block and ripple carry between blocks

Total 12 gate delay to complete c_{32}



The carry-out signals from four blocks: c_8, c_{16}, c_{24}, c_{32}

Figure 5.17. A hierarchical carry-lookahead adder with ripple-carry between blocks

Second-level Lookahead

$$g_i = x_i y_i$$

$$p_i = x_i + y_i$$

$$c_8 = g_7 + p_7 g_6 + p_7 p_6 g_5 + p_7 p_6 p_5 g_4 + p_7 p_6 p_5 p_4 g_3 + p_7 p_6 p_5 p_4 p_3 g_2 + p_7 p_6 p_5 p_4 p_3 p_2 g_1 + p_7 p_6 p_5 p_4 p_3 p_2 p_1 g_0 + p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0 c_0$$

$$P_0 = p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0$$

$$G_0 = g_7 + p_7 g_6 + p_7 p_6 g_5 + \cdots + p_7 p_6 p_5 p_4 p_3 p_2 p_1 g_0$$

$$c_8 = G_0 + P_0 c_0 \quad \text{(Block 0)}$$

$$c_{16} = G_1 + P_1 c_8 \quad \text{(Block 1)}$$

$$= G_1 + P_1 G_0 + P_1 P_0 c_0$$

Ripple carry between blocks

Second-level lookahead

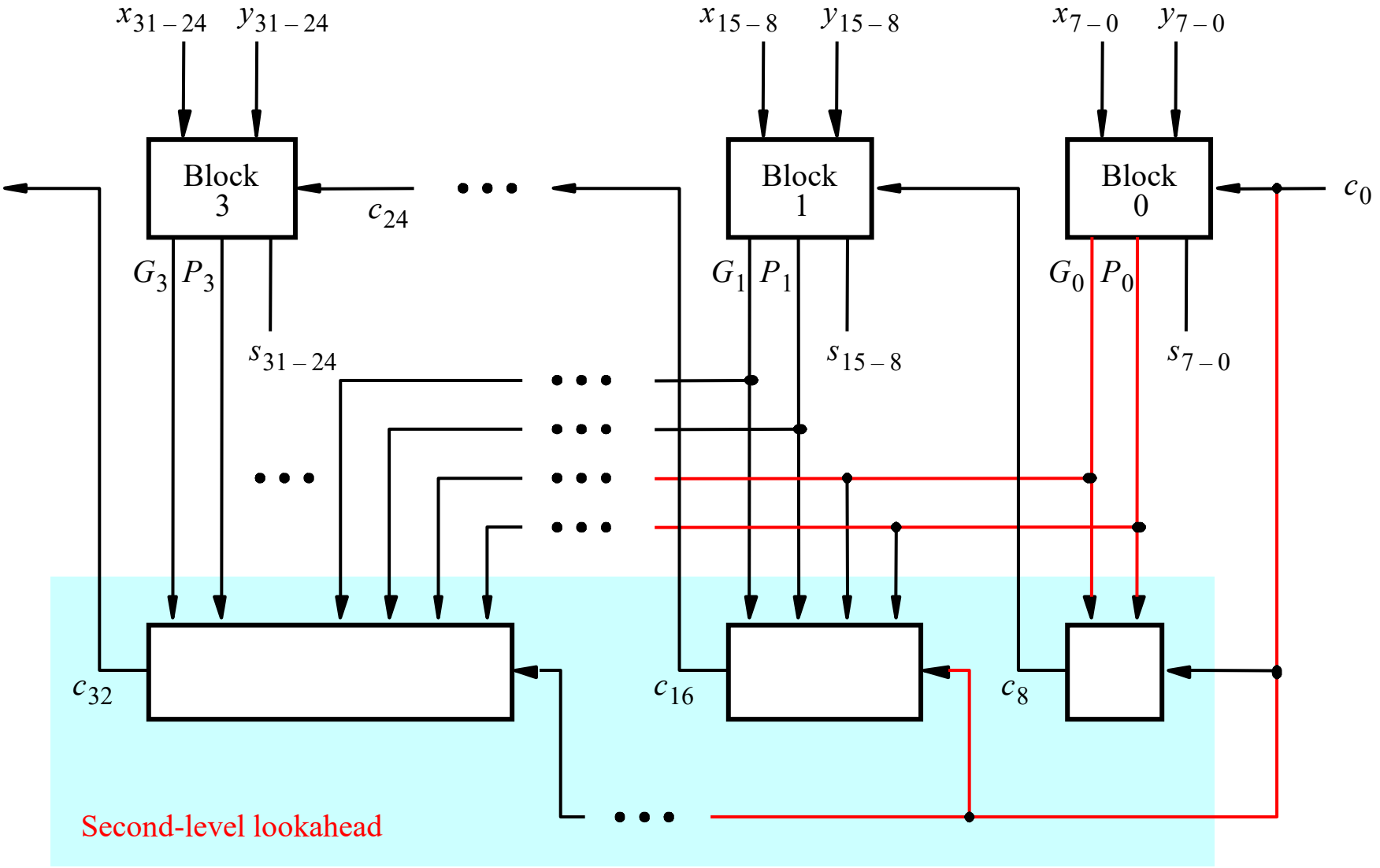
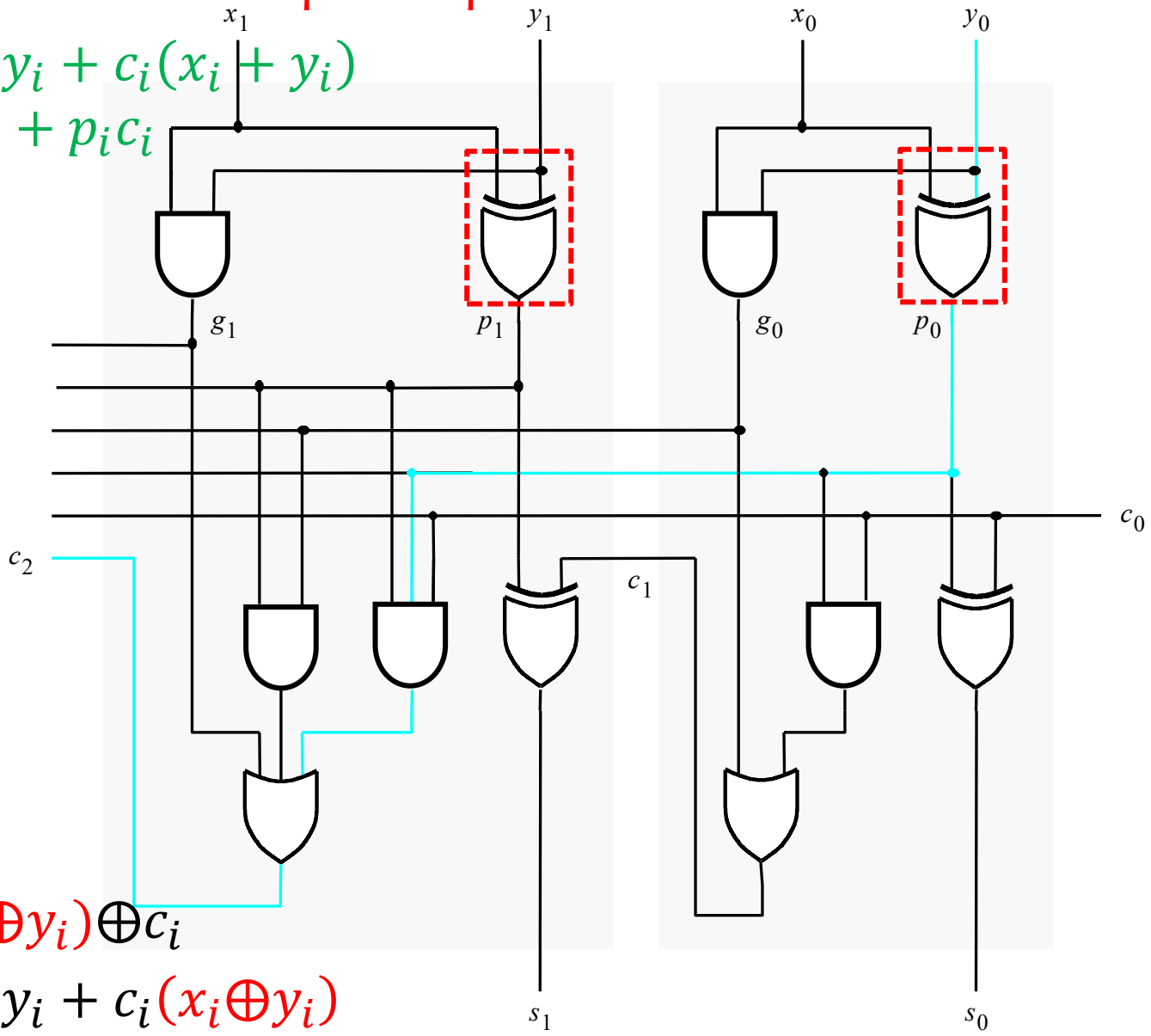


Figure 5.18. A hierarchical carry-lookahead adder.

Common units for multiple output

$$c_{i+1} = x_i y_i + c_i(x_i + y_i)$$
$$= g_i + p_i c_i$$



$$s_i = (x_i \oplus y_i) \oplus c_i$$

$$c_{i+1} = x_i y_i + c_i(x_i \oplus y_i)$$

Figure 5.19. An alternative design for a carry-lookahead adder.

Technology Considerations

Standard SOP : two level implementation

$$c_1 = g_0 + p_0 c_0$$
$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$
$$\vdots$$
$$c_8 = g_7 + p_7 g_6 + p_7 p_6 g_5 + p_7 p_6 p_5 g_4 + p_7 p_6 p_5 p_4 g_3 + p_7 p_6 p_5 p_4 p_3 g_2 + p_7 p_6 p_5 p_4 p_3 p_2 g_1 + p_7 p_6 p_5 p_4 p_3 p_2 p_1 g_0 + p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0 c_0$$

2-input AND, 3-input AND, 4-input AND, 5-input AND, 6-input AND, 7-input AND, 8-input AND, 9-input AND, 9-input OR

Cost =9 + (2+3+4+5+6+7+8+9+9)=62

Suppose that the maximum fan-in of the gates is four inputs.

Multilevel implementation

$$c_8 = (g_7 + p_7 g_6 + p_7 p_6 g_5 + p_7 p_6 p_5 g_4) + ((p_7 p_6 p_5 p_4)(g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0) + (p_7 p_6 p_5 p_4)(p_3 p_2 p_1 p_0) c_0$$

3 x 2-input AND, 3 x 3-input AND, 5 x 4-input AND, 2 x 3-input OR, 2 x 4-input OR

Cost =15 + (2x3+3x3+5x4+2x3+2x4)=64

MULTIPLICATION

Multiplication

$n\ bit + n\ bit = n + 1\ bit$

$n\ bit \times n\ bit = 2n\ bit$

Multiplicand M	(14)	1 1 1 0
Multiplier Q	(11)	x 1 0 1 1
		1 1 1 0
		1 1 1 0
		0 0 0 0
		1 1 1 0
Product P	(154)	1 0 0 1 1 0 1 0

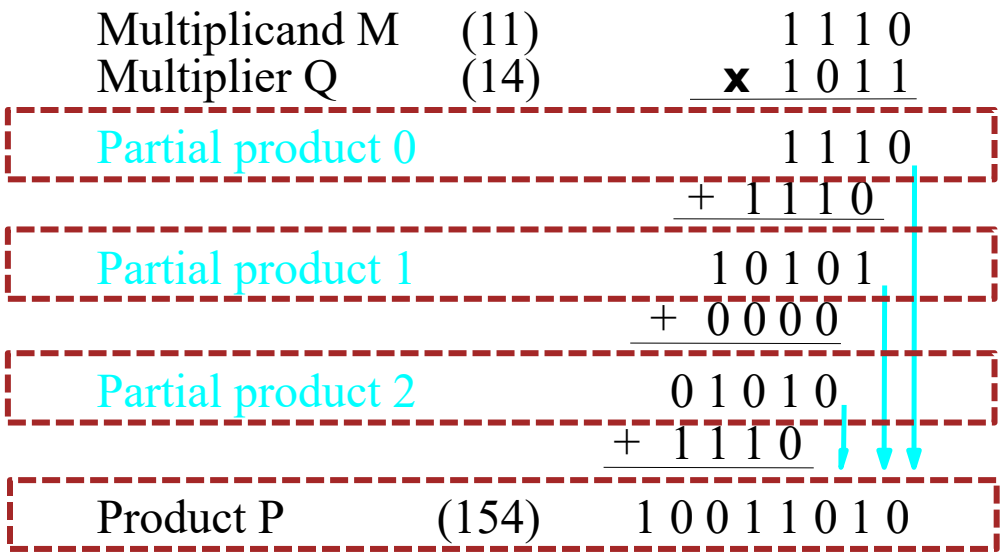
$$M \times Q = P$$

$$m_3 m_2 m_1 m_0 \times q_3\ q_2\ q_1\ q_0 = p_7\ p_6\ p_5\ p_4\ p_3\ p_2\ p_1\ p_0$$

(a) Multiplication by hand

Figure 5.31. Multiplication of unsigned numbers.

Multiplication



(b) Multiplication for implementation in hardware

Figure 5.31. Multiplication of unsigned numbers.

Multiplication

$PP0 = \quad m_3q_0 \quad m_2q_0 \quad m_1q_0 \quad m_0q_0$

$PP0: \quad 0 \quad pp0_3 \quad pp0_2 \quad pp0_1 \quad pp0_0$

$+ \quad m_3q_1 \quad m_2q_1 \quad m_1q_1 \quad m_0q_1 \quad 0$

$PP1: \quad pp1_4 \quad pp1_3 \quad pp1_2 \quad pp1_1 \quad \boxed{pp1_0} \quad p_0$

$PP1: \quad 0 \quad pp1_4 \quad pp1_3 \quad pp1_2 \quad pp1_1$

$+ \quad m_3q_2 \quad m_2q_2 \quad m_1q_2 \quad m_0q_2 \quad 0$

$PP2: \quad pp2_4 \quad pp2_3 \quad pp2_2 \quad pp2_1 \quad \boxed{pp2_0} \quad p_1$

$PP2: \quad 0 \quad pp2_4 \quad pp2_3 \quad pp2_2 \quad pp2_1$

$+ \quad m_3q_3 \quad m_2q_3 \quad m_1q_3 \quad m_0q_3 \quad 0$

$PP3: \quad c_{out} \quad pp3_4 \quad pp3_3 \quad pp3_2 \quad pp3_1 \quad pp3_0$

$P: \quad p_7 \quad p_6 \quad p_5 \quad p_4 \quad p_3 \quad p_2$

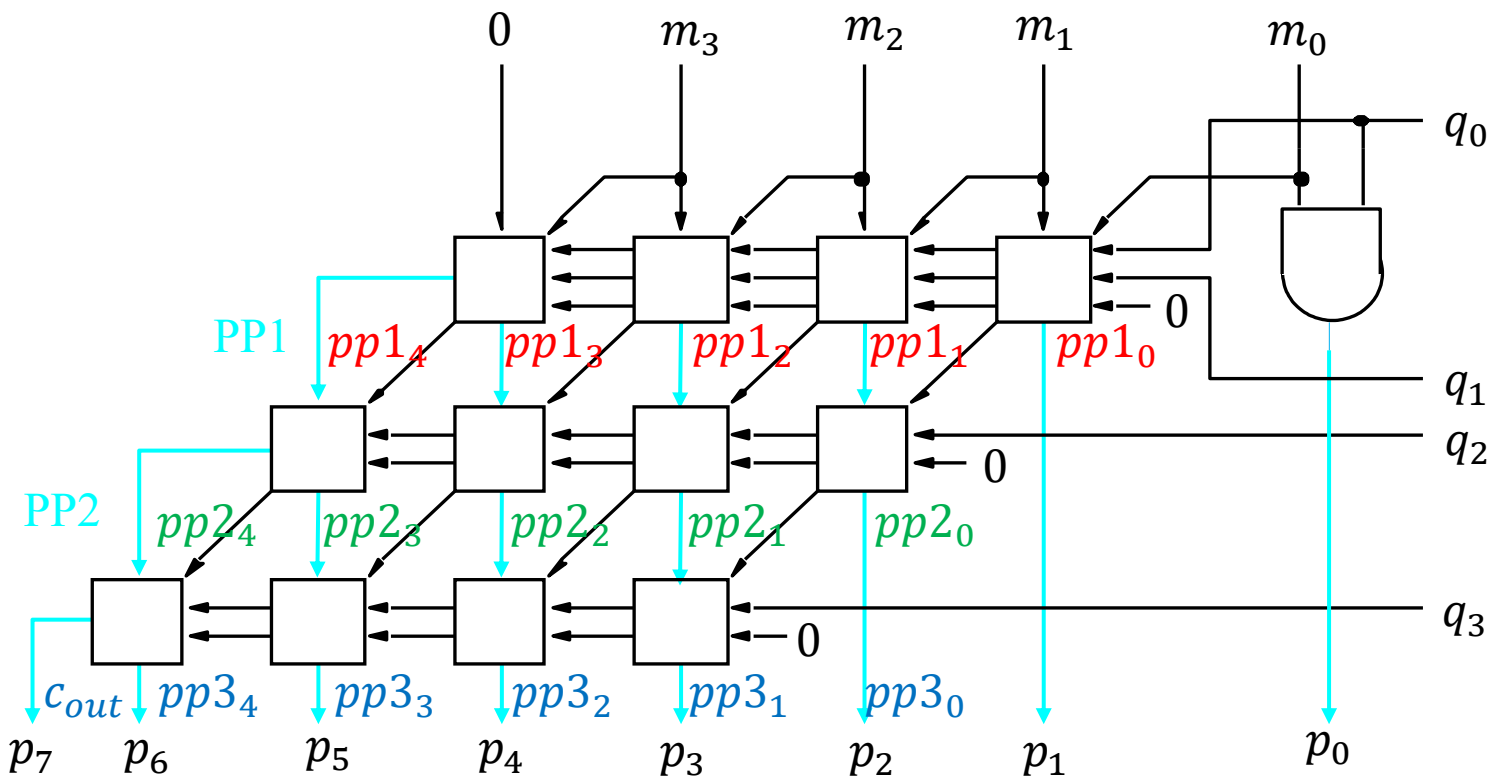
$pp0_0 = pp1_0 = p_0$

$pp1_1 = pp2_0 = p_1$

$pp2_1 = pp3_0 = p_2$

Figure 5.31. Multiplication of unsigned numbers.

4 x 4 multiplier circuit

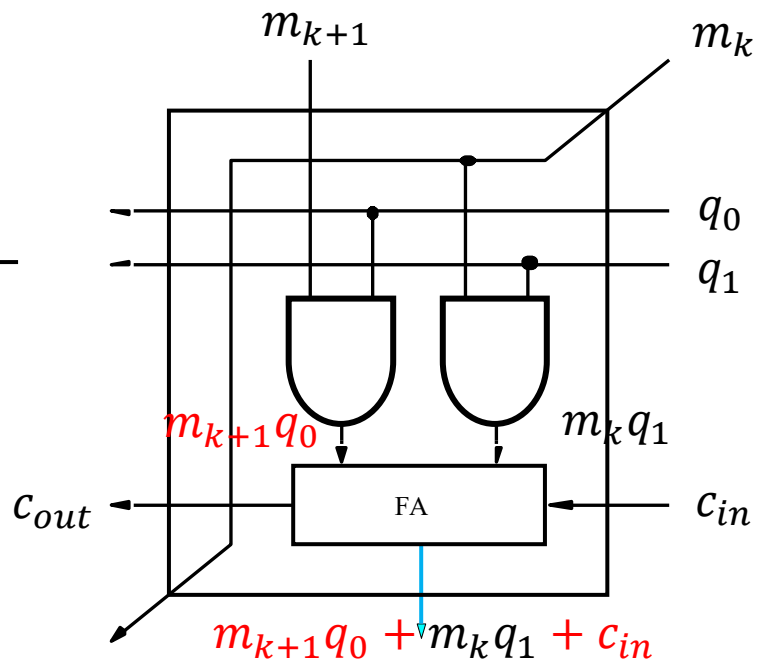


(a) Structure of the circuit

Figure 5.32. A 4 x 4 multiplier circuit.

4 x 4 multiplier circuit

$$
\begin{array}{lccccc}
PP0: & 0 & m_3q_0 & m_2q_0 & m_1q_0 & m_0q_0 \\
+ & m_3q_1 & m_2q_1 & m_1q_1 & m_0q_1 & 0 \\
\hline
PP1: & pp1_4 & pp1_3 & pp1_2 & pp1_1 & pp1_0
\end{array}
$$



(b) A block in the top row

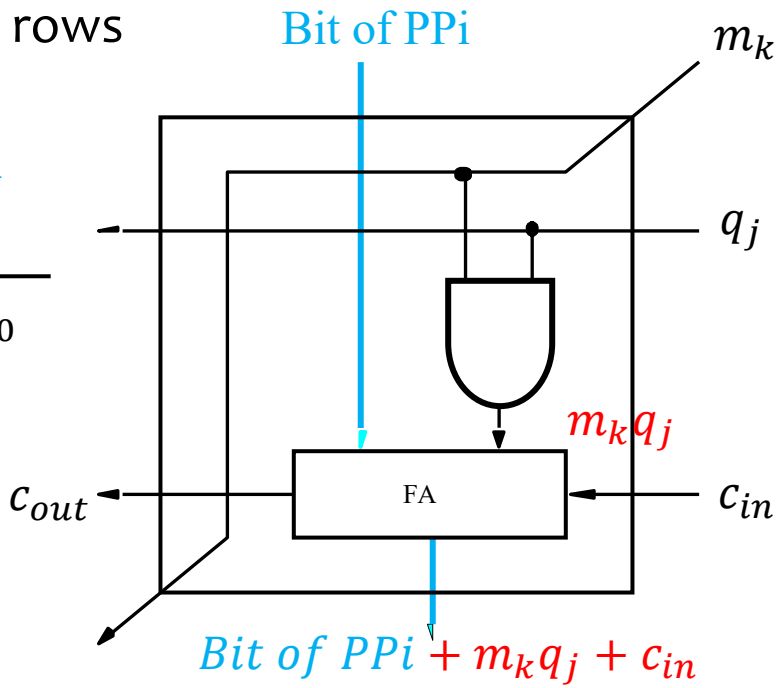
Module for PP1

Figure 5.32. A 4 x 4 multiplier circuit.

4 x 4 multiplier circuit

First row $j = 2$ in a block in the bottom two rows

$PP1$:		0	$pp1_4$	$pp1_3$	$pp1_2$	$pp1_1$
	+	m_3q_2	m_2q_2	m_1q_2	m_0q_2	0
$PP2$:		$pp2_4$	$pp2_3$	$pp2_2$	$pp2_1$	$pp2_0$

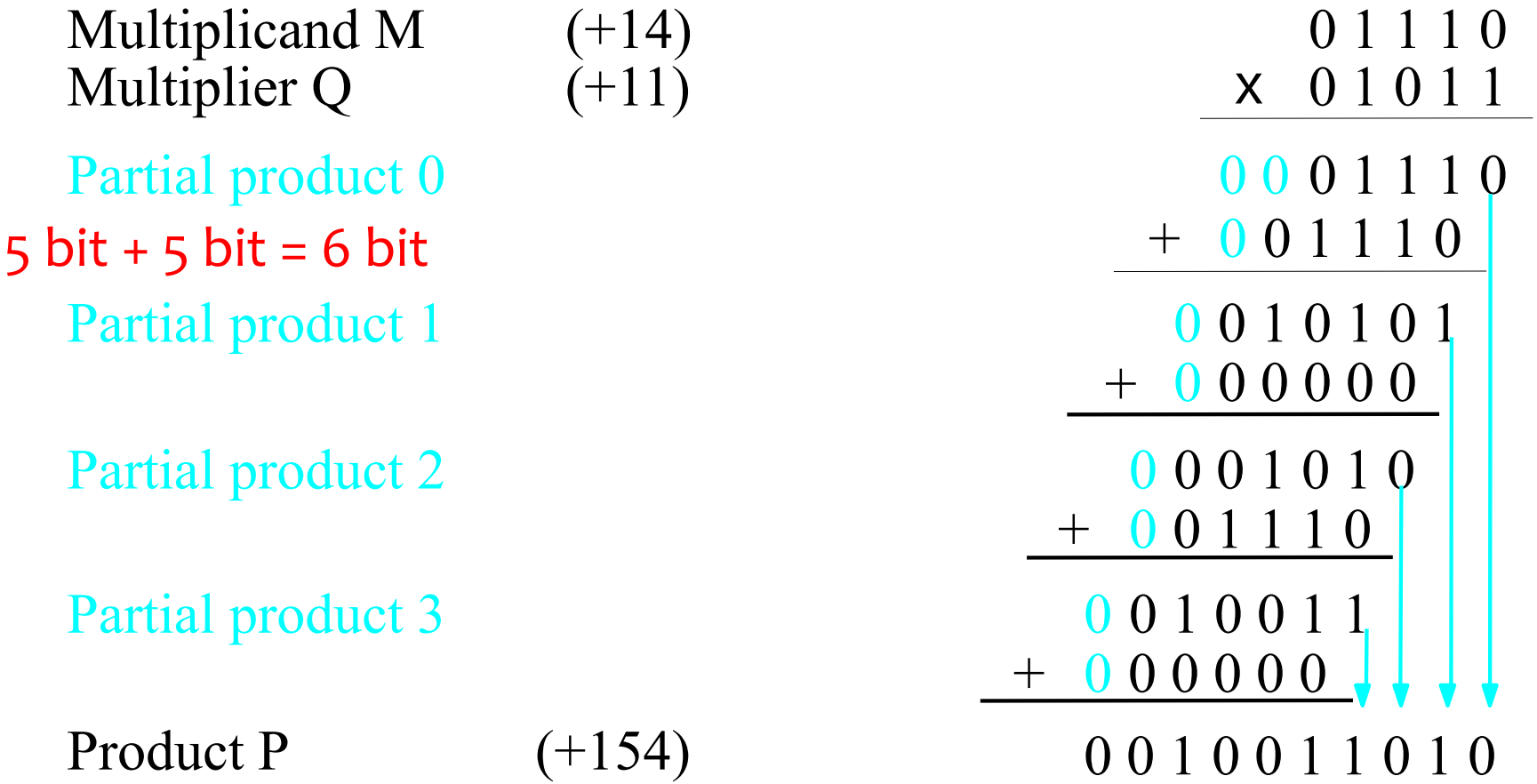


(c) A block in the bottom two rows

Module for PP2 and P

Figure 5.32. A 4 x 4 multiplier circuit.

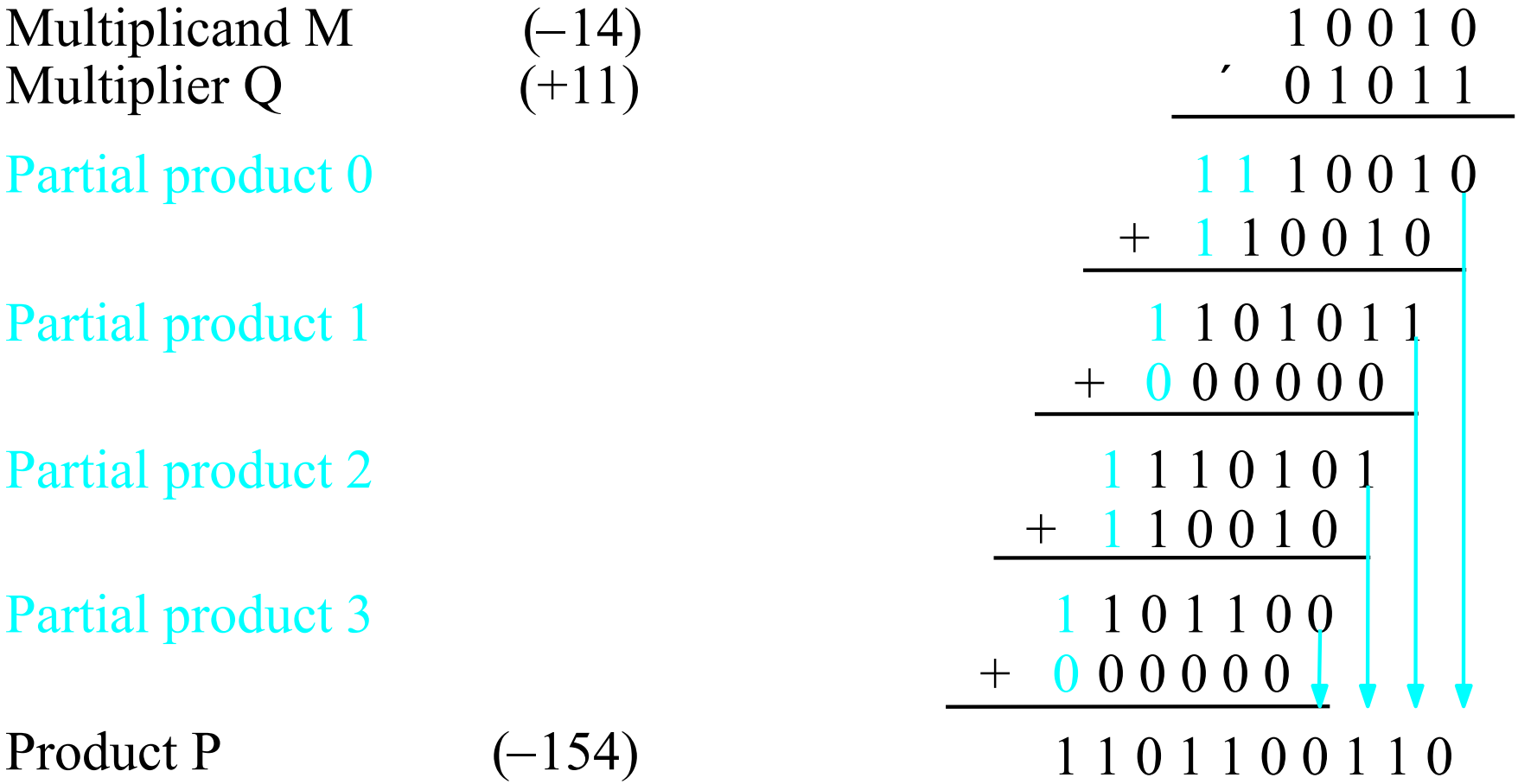
Multiplication of signed numbers



(a) Positive multiplicand

Figure 5.33. Multiplication of signed numbers.

Multiplication of signed numbers



(b) Negative multiplicand

Figure 5.33. Multiplication of signed numbers.

OTHER NUMBER REPRESENTATIONS

Fixed-Point Numbers

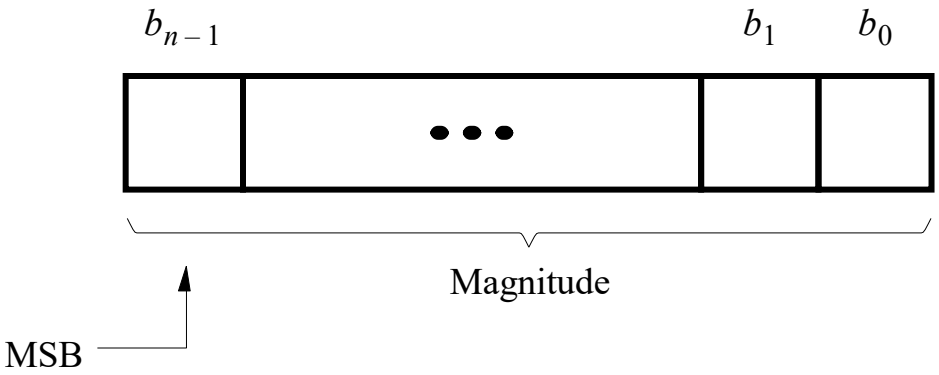
$$B = b_{n-1}b_{n-2}\cdots b_1 b_0 . b_{-1}b_{-2}\cdots b_{-k}$$

$$V(B) = \sum_{i=-k}^{n-1} b_i \times 2^i$$

The position of radix point is assumed; hence the name fixed-point number.

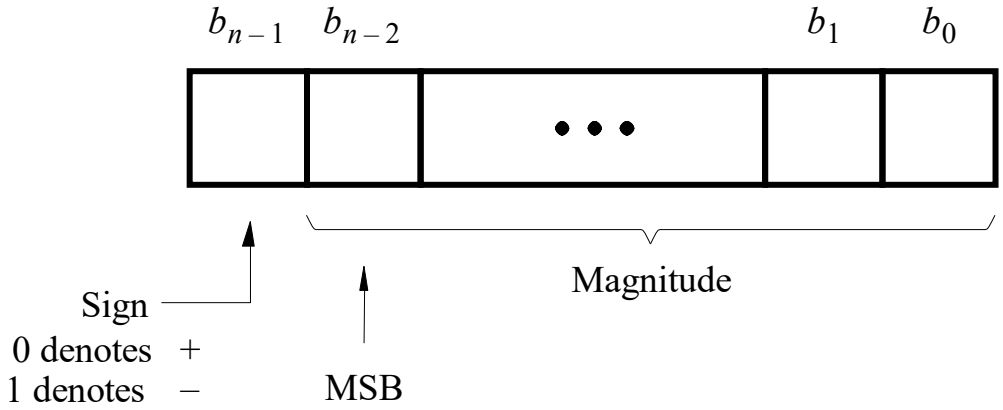
If the radix point is not shown, then it is assumed to be to the right of the least-significant digit, which means that the number is an integer.

Fixed-Point Numbers



Range: o ~$2^n - 1$

(a) Unsigned number

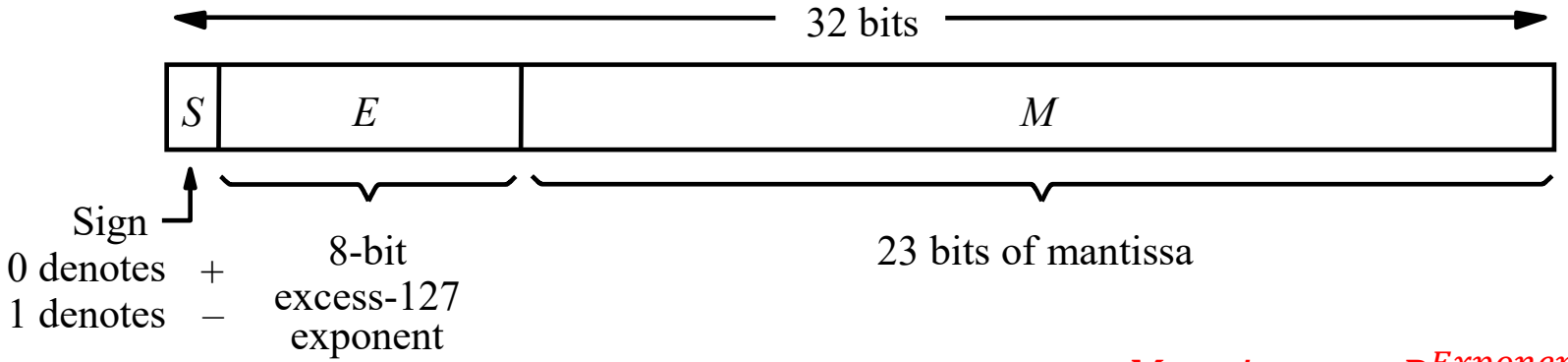


Range: -2^{n-1} ~$2^{n-1} - 1$

(b) Signed number

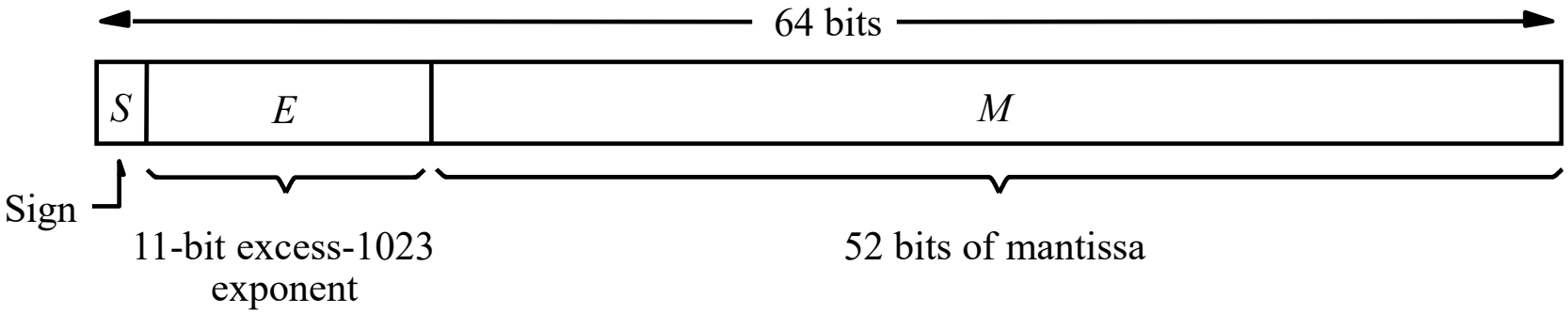
Floating-Point Numbers

Using excess-127 is convenient for adding and subtracting floating-point number



$Mantissa \times R^{Exponent}$

(a) Single precision



(c) Double precision

Figure 5.34. IEEE Standard floating-point formats.

Floating-Point Numbers

- Single-Precision Floating-Point Format
  - Exponent = E – 127
    - Range: 0~255, 0: real "0", 255: "∞"
  - $Value = \pm 1.M \times 2^{E-127}$
  - The exponent range of 2^{-126} to 2^{127} corresponds to about $10^{\pm 38}$
- Double-Precision Floating-Point Format
  - Exponent = E – 1023
  - $Value = \pm 1.M \times 2^{E-1023}$
  - The exponent range of 2^{-1022} to 2^{1023} corresponds to about $10^{\pm 308}$

Floating-Point Numbers

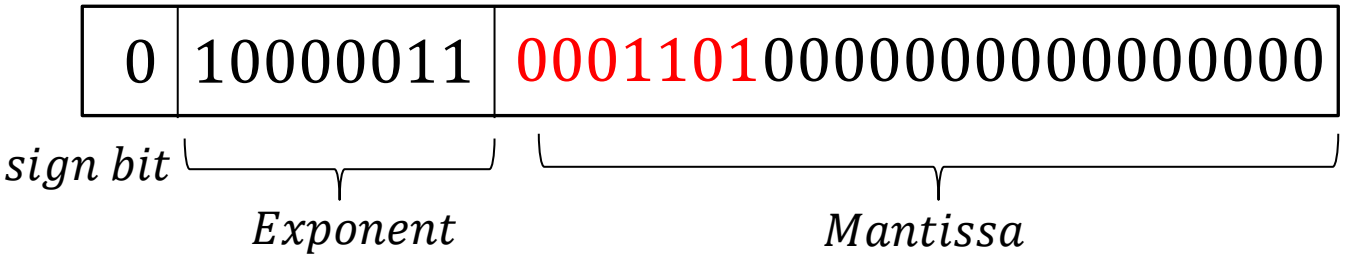
- Single-Precision Floating-Point Format
  - $Value = \pm 1.M \times 2^{E-127}$

$(17.625)_{10} = (10001.101)_2$

$10001.101 = 1.0001101 \times 2^4$

$Mantissa = 00011010000000000000000$

$E = 4 + 127 = 10000011 = (131)_{10}$



Floating-Point Numbers

- Single-Precision Floating-Point Format
  - $Value = \pm 1.M \times 2^{E-127}$
  - Exponent = E – 127
  - Range of Exponent : $-2^7 \sim 2^7 - 1 = -128 \sim 127$
  - Range of E : Exponent + 127 $= -1 \sim 254 = 0 \sim 255$
  - $(-1)_{10} = (11111111)_2 = (255)_{10}$

Binary-Coded-Decimal Representation

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

Don't care term
$1010, 1011, 1100,$
$1101, 1110, 1111$

$$(13)_{10} = (1101)_2 = (D)_{16}$$

$$= (0001)_2(0011)_2$$

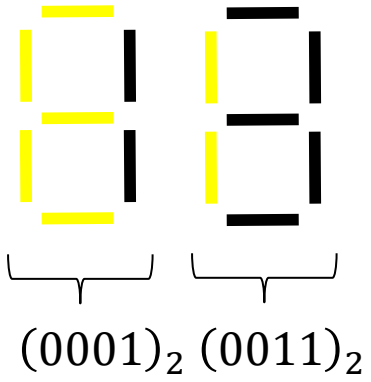


Table 5.3. Binary-coded decimal digits.

Addition of BCD digits

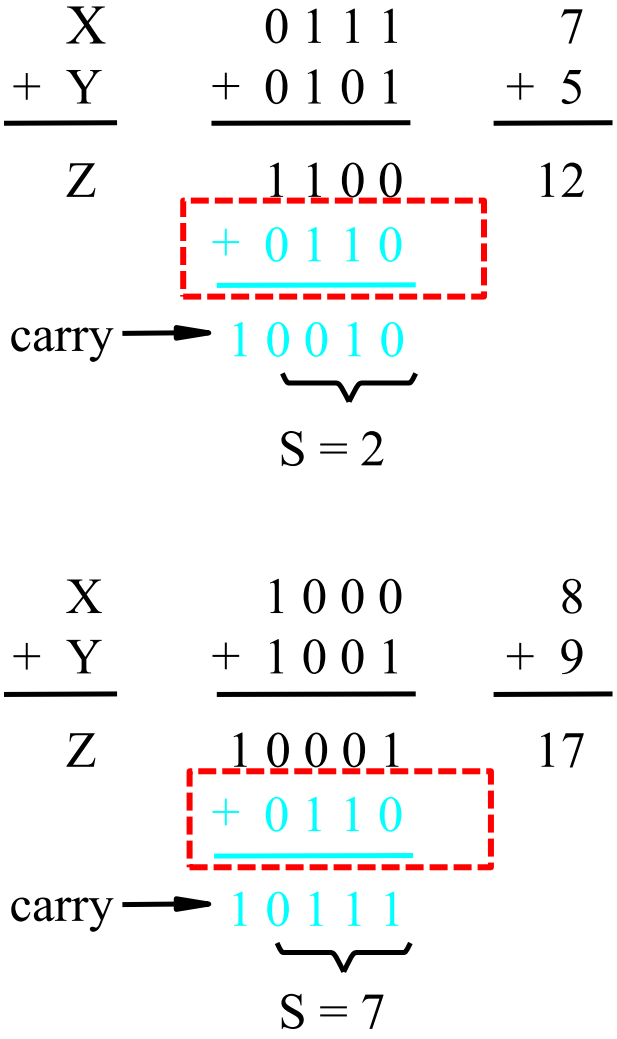


Figure 5.35. Addition of BCD digits.

BCD adder

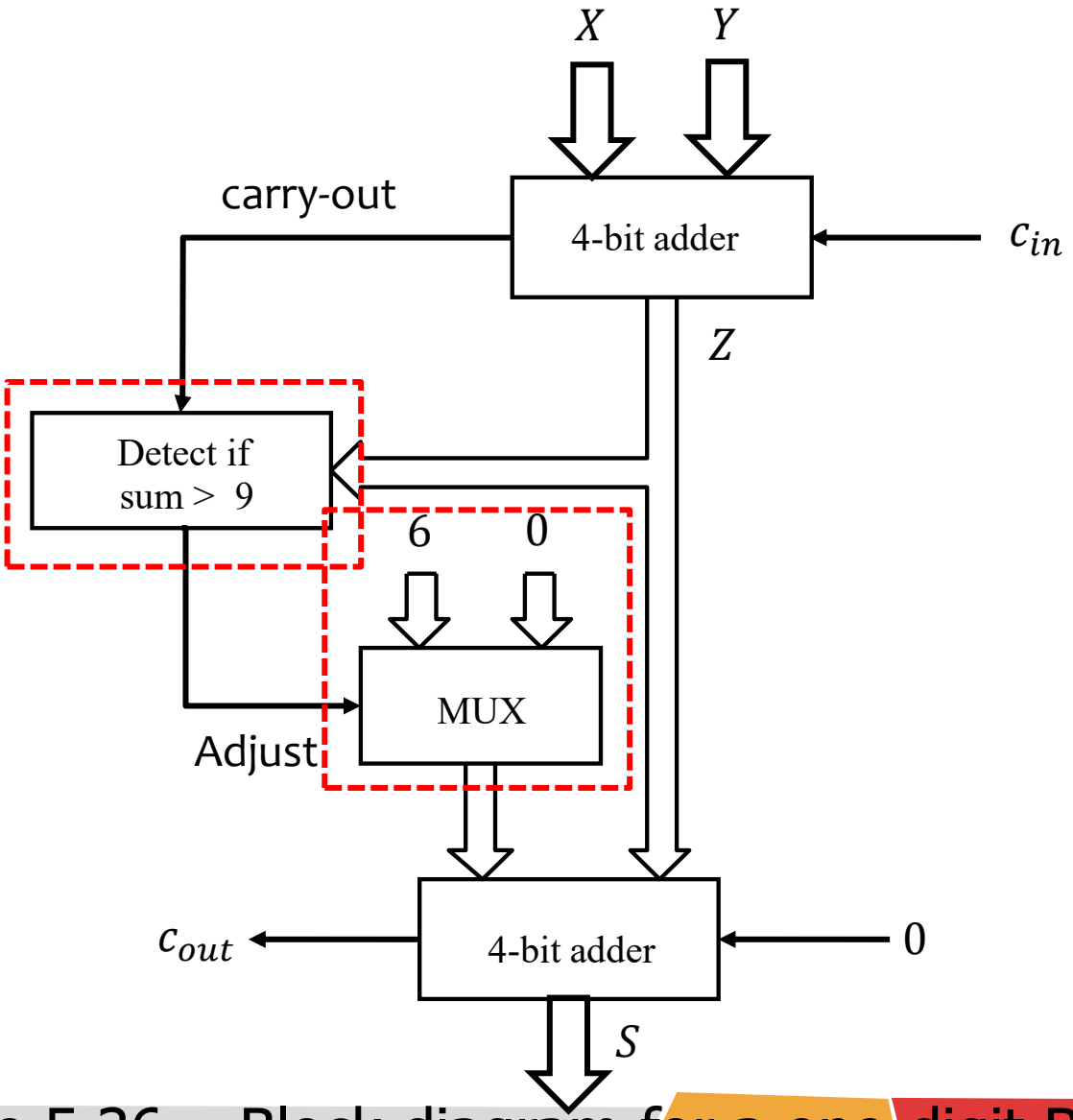
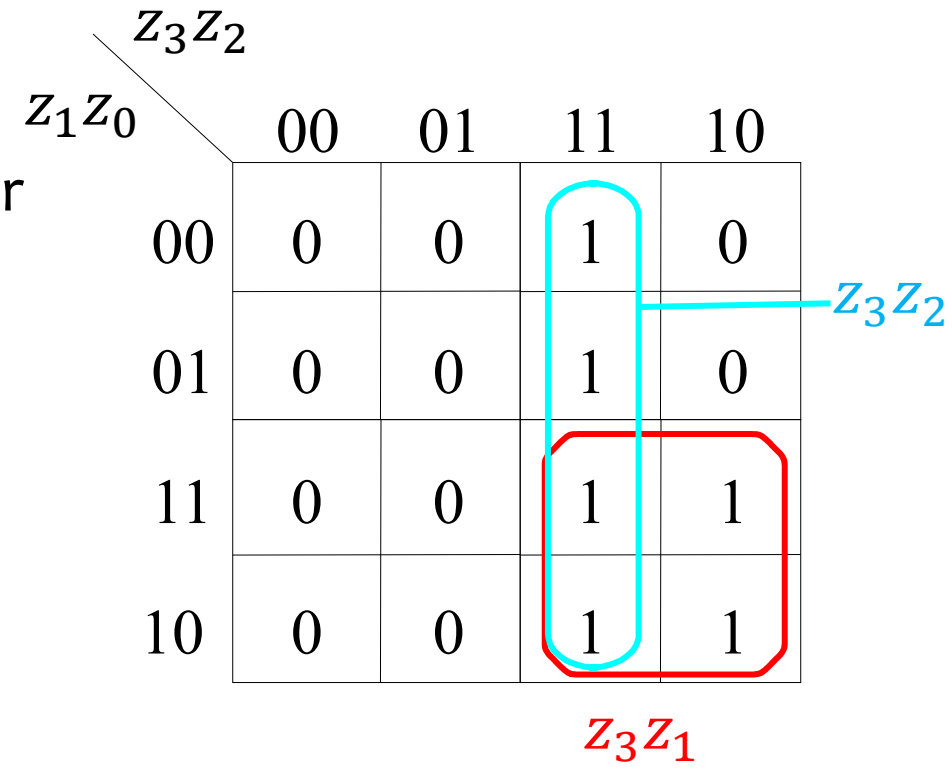


Figure 5.36. Block diagram for a one-digit BCD adder.

Detect if sum > 9

(1) If carry out of first 4-bit adder is occurred, i.e. $c_4 = 1$

(2) If sum of first 4-bit adder $Z > 1, z_3 z_2 z_1 z_0 > 1001$



$$c_{out} = c_4 + z_3 z_2 + z_3 z_1$$

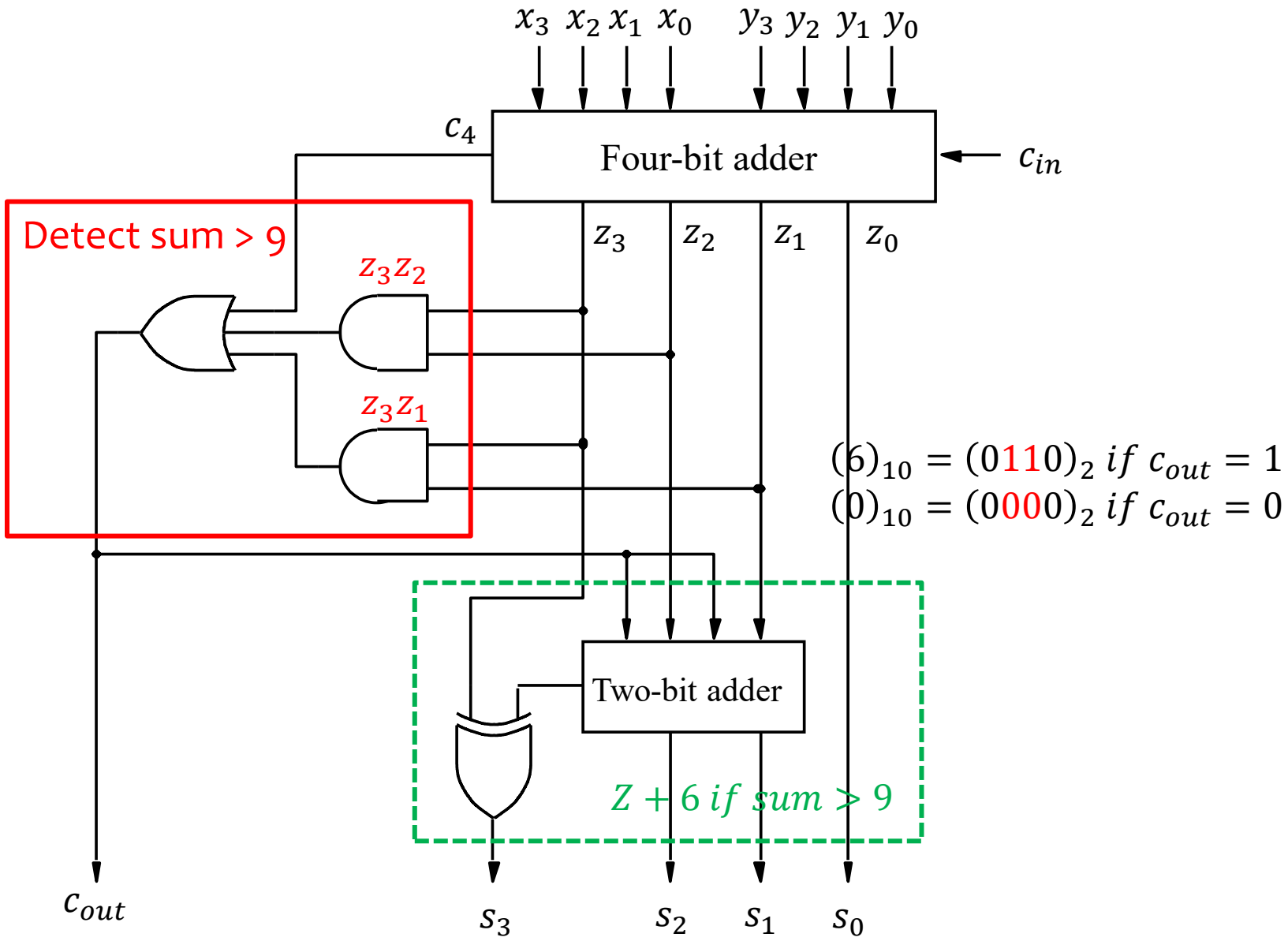


Figure 5.39. Circuit for a one-digit BCD adder.

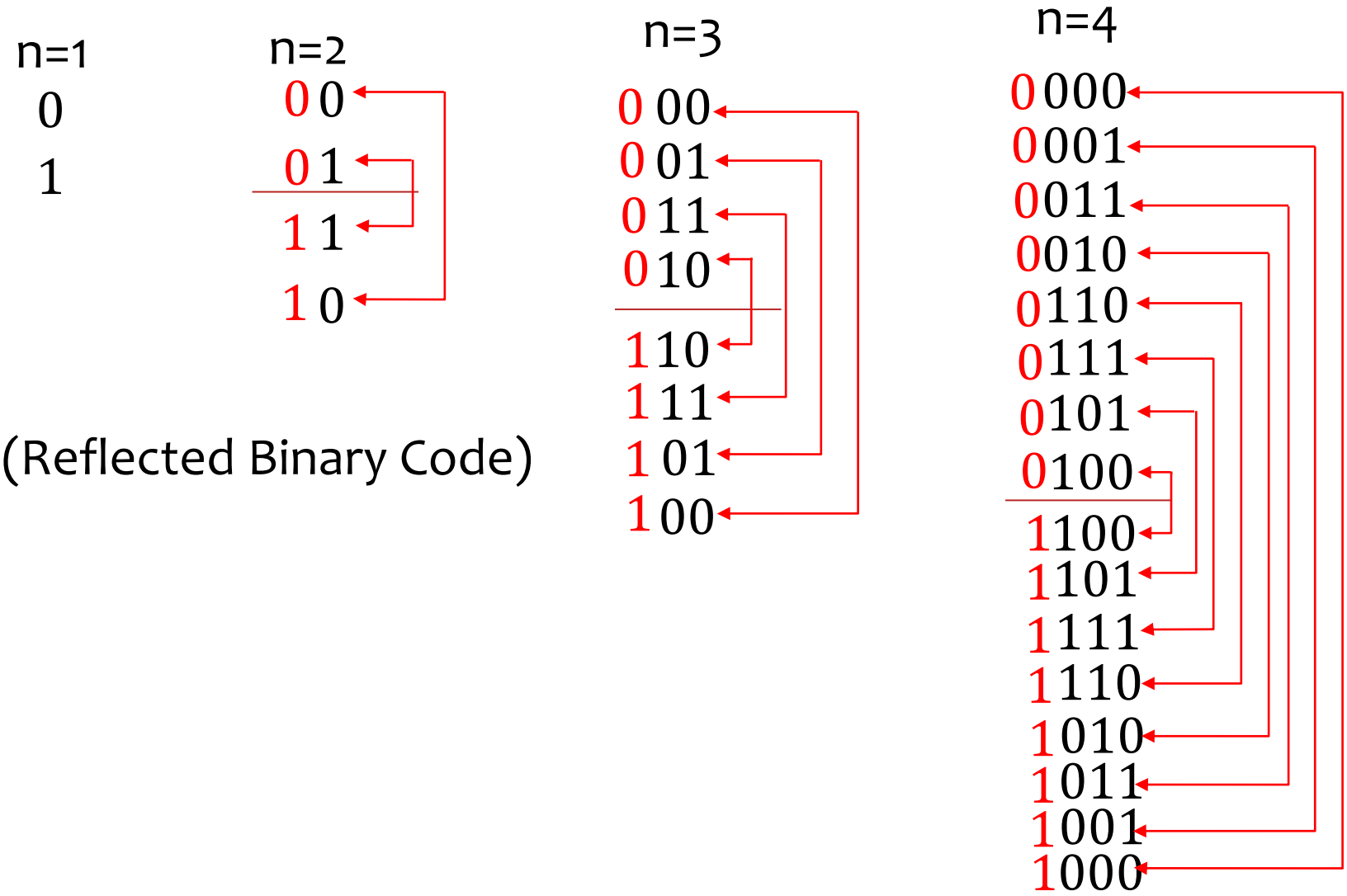
Gray Code

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

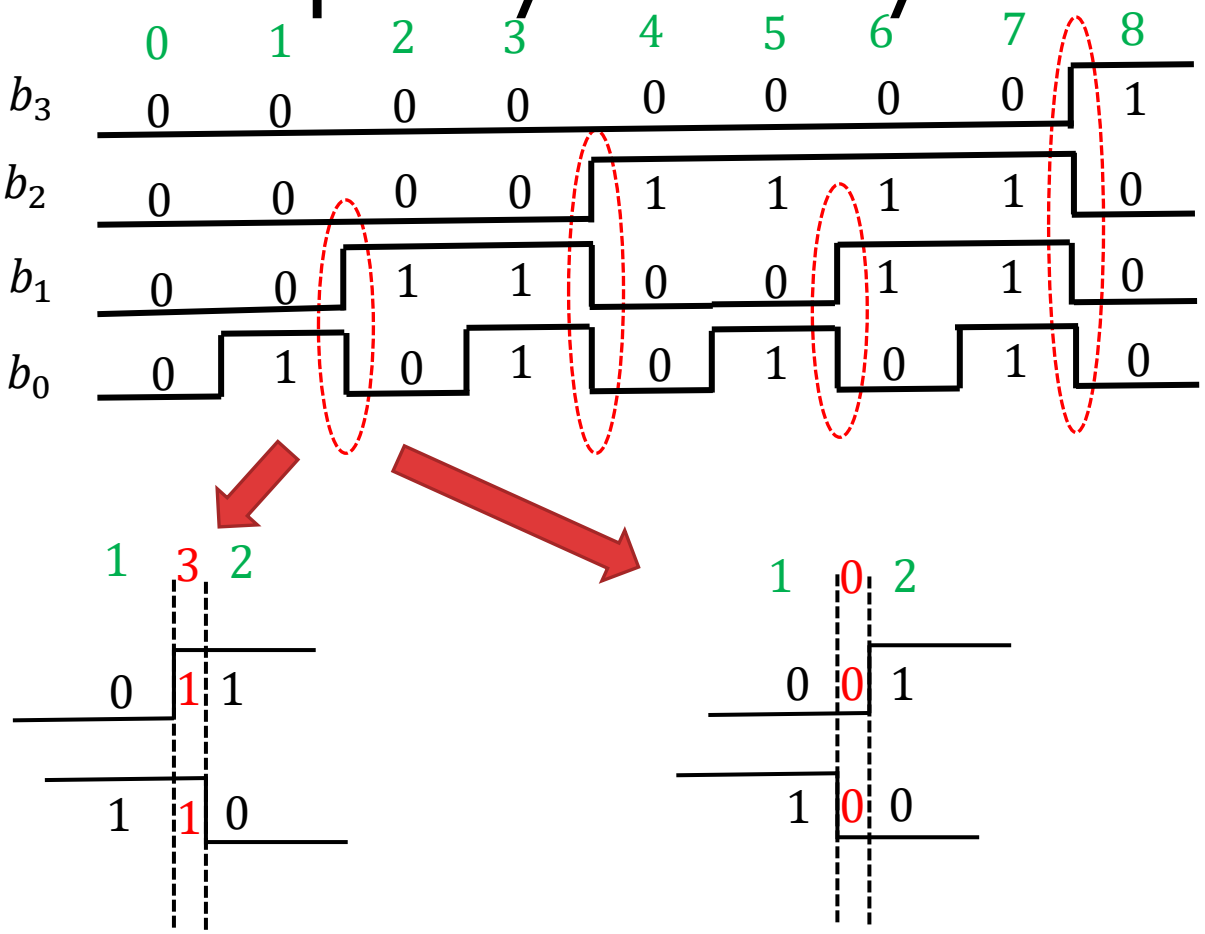
Successive Gray code has 1 bit difference only

Generation of Gray Code



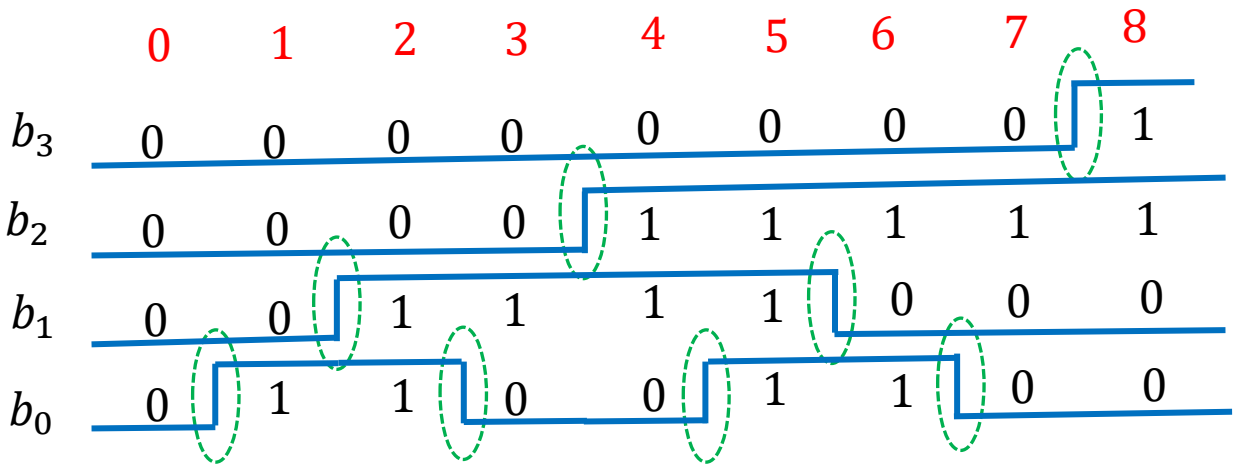
RBC (Reflected Binary Code)

Property of Gray Code



Binary Counter

Property of Gray Code



ASCII CHARACTER CODE

128 characters
English only

Bit positions	Bit positions 654							
3210	000	001	010	011	100	101	110	111
0000	NUL	DLE	SPACE	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	\	1	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	—	o	DEL

NUL	Null/Idle	SI	Shift in
SOH	Start of header	DLE	Data link escape
STX	Start of text	DC1-DC4	Device control
ETX	End of text	NAK	Negative acknowledgement
EOT	End of transmission	SYN	Synchronous idle
ENQ	Enquiry	ETB	End of transmitted block
ACQ	Acknowledgement	CAN	Cancel (error in data)
BEL	Audible signal	EM	End of medium
BS	Back space	SUB	Special sequence
HT	Horizontal tab	ESC	Escape
LF	Line feed	FS	File separator
VT	Vertical tab	GS	Group separator
FF	Form feed	RS	Record separator
CR	Carriage return	US	Unit separator
SO	Shift out	DEL	Delete/Idle

Bit positions of code format = | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Table 5.4. The seven-bit ASCII code.

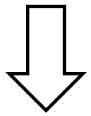
Parity

- Error-Detecting Code (parity checker)

	$b_6b_5b_4b_3b_2b_1b_0$	With even parity	With odd parity
ASCII A =	1 0 0 0 0 0 1	0 1000001	1 1000001
ASCII T =	1 0 1 0 1 0 0	1 1010100	0 1010100
		P	P

- Even Parity

$$P = b_6 \oplus b_5 \oplus b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus b_0$$



$$C = P \oplus b_6 \oplus b_5 \oplus b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus b_0$$

Parity Checker

- Even Parity

Case of Character ‘A’

$$P = b_6 \oplus b_5 \oplus b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus b_0$$
$$= 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 0$$

No Error

$$C = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 0$$

Error occurred at b_4

$$C = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1$$

Binary Code for Text

- ASCII (American Standard Code for Information Interchange)
  - English only
  - How can we represent the other languages such as Hangeul, Chinese character, etc?
- Uni-Code
  - Unicode Consortium
  - UTF (Unicode Transformation Format) : UTF-8, UTF-16, UTF-32
  - UCS (Universal Coded Character Sets) : UCS-2 (16 bit), UCS-4 (32 bit)
- EBCDIC (Extended Binary Coded Decimal Interchange Code)
  - eight-bit character encoding used mainly on IBM mainframe and IBM midrange computer operating systems

EXAMPLES OF SOLVED PROBLEMS

Example 5.7

Conversion the decimal number into a hexadecimal number

Convert $(14959)_{10}$

			Remainder	Hex digit	
14959 ÷ 16	=	934	15	F	LSB
934 ÷ 16	=	58	6	6	
58 ÷ 16	=	3	10	A	
3 ÷ 16	=	0	3	3	MSB

Result is $(3A6F)_{16}$

Figure 5.40. Conversion from decimal to hexadecimal.

Example 5.8

Convert $(0.8254)_{10}$

fractional part : successive multiplication by 2

$0.8254 \times 2 = 1.6508$ → 1 MSB

$0.6508 \times 2 = 1.3016$ → 1

$0.3016 \times 2 = 0.6032$ → 0

$0.6032 \times 2 = 1.2064$ → 1

$0.2064 \times 2 = 0.4128$ → 0

$0.4128 \times 2 = 0.8256$ → 0

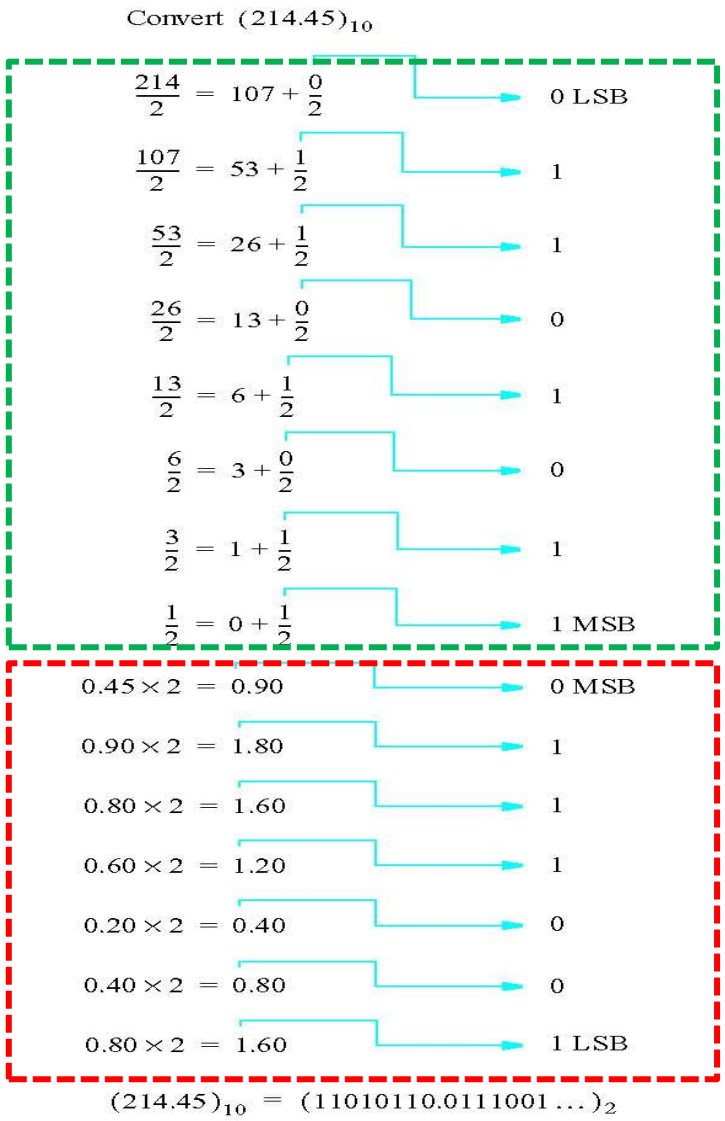
$0.8256 \times 2 = 1.6512$ → 1

$0.6512 \times 2 = 1.3024$ → 1 LSB

$$(0.8254)_{10} = (0.11010011\ldots)_2$$

Figure 5.41. Conversion of fractions from decimal to binary.

Example 5.9



Integer part : successive division by 2

fractional part : successive multiplication by 2

Figure 5.42. Conversion of fixed point numbers from decimal to binary.

Example 5.10

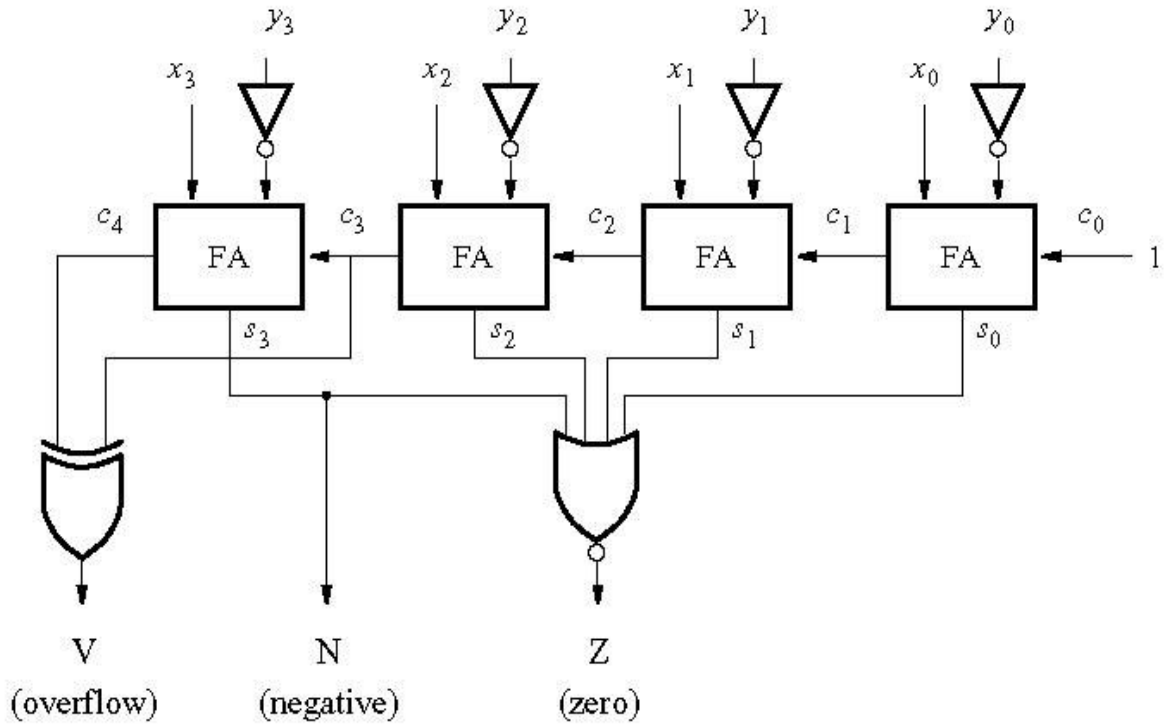


Figure 5.43. A comparator circuit.

$$X = x_3x_2x_1x_0$$
$$Y = y_3y_2y_1y_0$$

$$X = x_3x_2x_1x_0$$
$$-Y = y_3'y_2'y_1'y_0'$$
$$+1$$

$$c_4S = c_4s_3s_2s_1s_0$$

$$S = s_3s_2s_1s_0$$
$$V = c_4 \oplus c_3$$
$$N = s_3$$
$$Z = (s_3 + s_2 + s_1 + s_0)'$$
$$= \begin{cases} 1 \; if \; X - Y = 0 \\ 0 \quad otherwise \end{cases}$$

Example 5.10

```
if (X>Y)
        F=X;
else
        F=Y;
```

$$Z = \begin{cases} 1 \; if \; X = Y \\ 0 \; if \; X \neq Y \end{cases}$$

$$N = \begin{cases} 1 \; if \; X < Y \\ 0 \; if \; X \geq Y \end{cases}$$

$$P = N'Z' = \begin{cases} 1 \; if \; X > Y \\ 0 \; if \; X \leq Y \end{cases}$$

Summary

- The circuit, which implement the addition of two bits, is called a half adder.
- A full adder generates sum and carry out by using the carry in bit as well as two input bits.
- An n-bit ripple carry adder can be easily implemented by connecting full adders.
- The delay of an n-bit ripple carry adder depends on the size of the numbers, that is, $Delay \approx n\Delta t$, Δt is the one bit gate delay.
- Subtraction can be carried out by adder and the complement.

Summary

- Arithmetic overflow can be detected by carry out bits.
- To improve the performance of an ripple carry adder, a carry-lookahead adder can be used as a fast adder.
- Multiplication can be implemented by adder, shift operation and memory.
- Hardware multiplier accelerates the speed of multiplication.