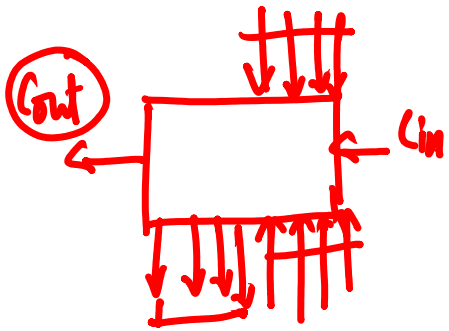
ADDER CIRCUIT

- Parallel adder/subtractor using 2's complement system:

1. Why does C_0 have to be a 1 in order to use the adder circuit in Figure 6-13 as a subtractor?
2. Assume that $[A] = 0011$ and $[B] = 0010$ in Figure 6-14. If ADD = 1 and SUB = 0, determine the logic levels at the OR gate outputs.
3. Repeat question 2 for ADD = 0, SUB = 1.
4. *True or false:* When the adder/subtractor circuit is used for subtraction, the 2's complement of the subtrahend appears at the input of the adder.

A = 0011 → (3)
B = 0010 → (2)
→ A+B, A−B

Add=1, SUB=0, A+B

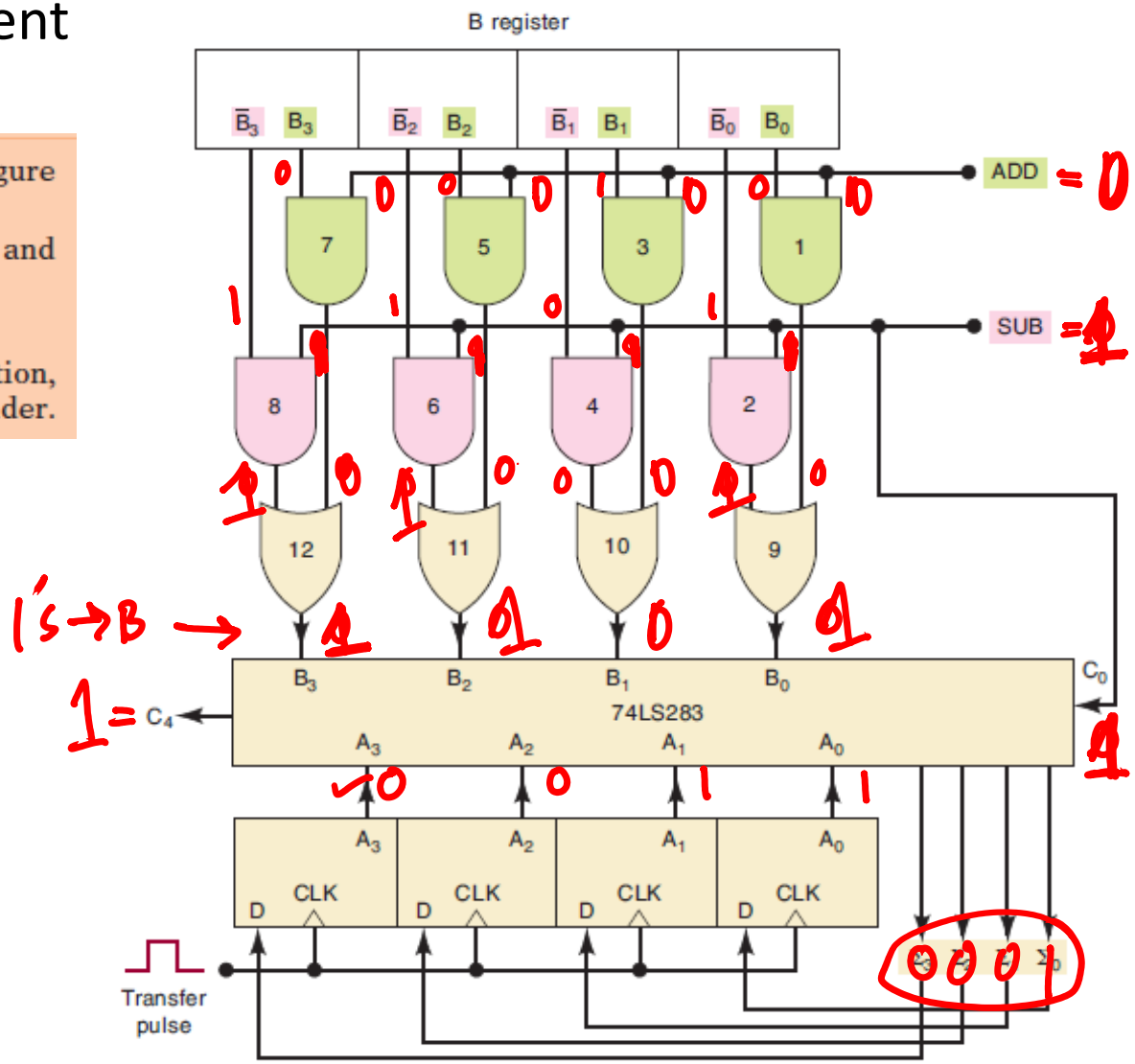


0010
+ 0011
0101 → (5)

Add=0, SUB=1 B=0010

1's → B = 1101

1101
+ 1
1 0001



Ref (1): pp 397-398

ADDER CIRCUIT

- Parallel adder/subtractor using 2's complement system:

M=0, Z=B → Addition.
M=1, Z=$\overline{B}$ → 1's complement.

A = 0011 → (3)
B = 0010 → (2) 1's → B = 1101

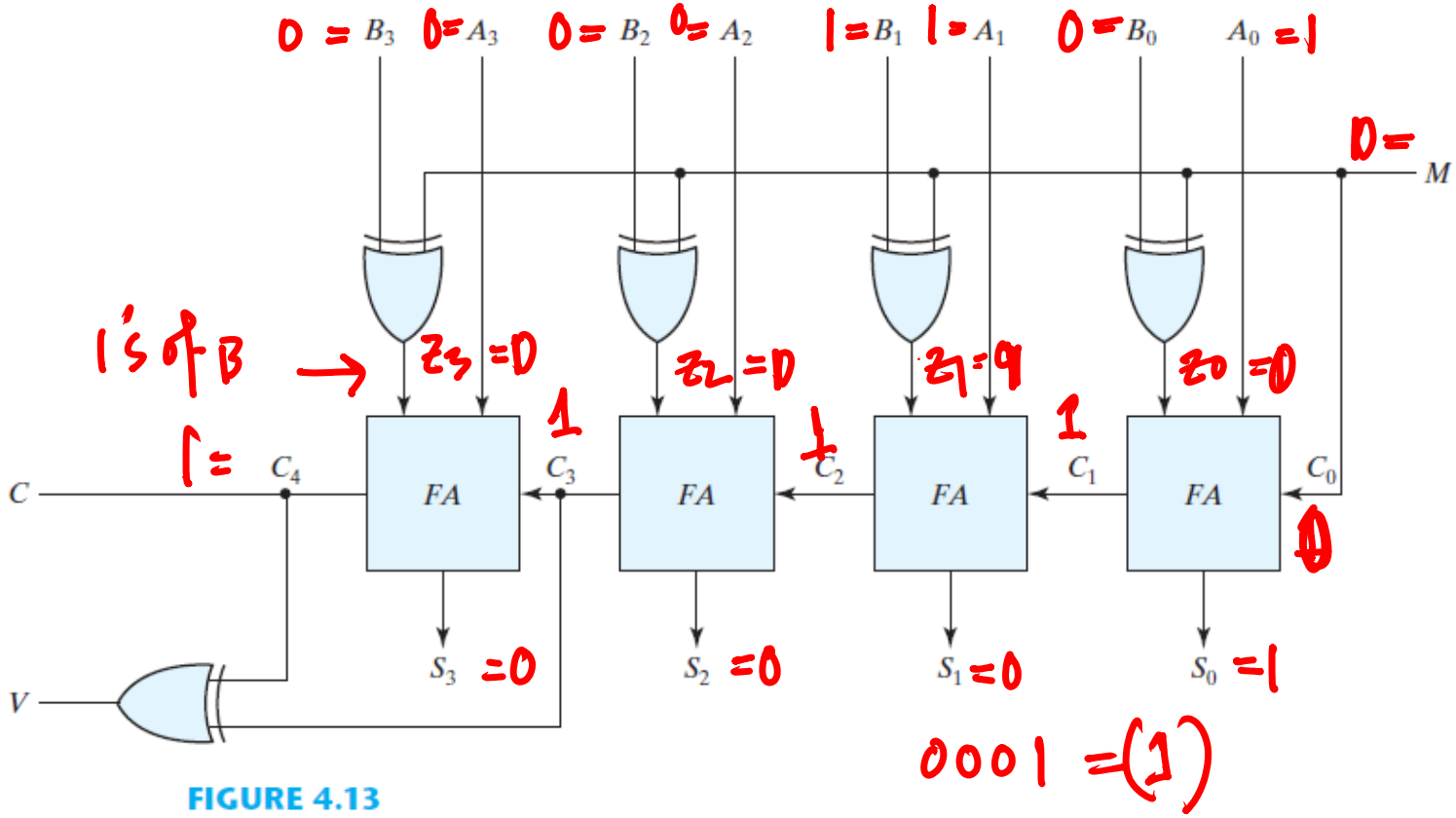
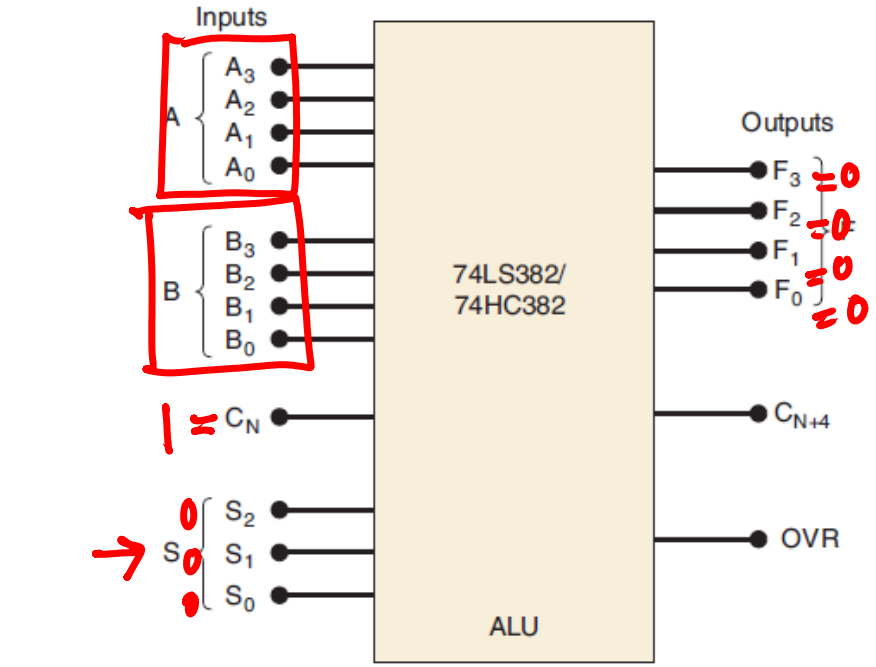


FIGURE 4.13
Four-bit adder–subtractor (with overflow detection)

Ref (2): pp 182

ADDER CIRCUIT

- ALU Operation:



A = 4-bit input number
B = 4-bit input number
C_N = carry into LSB position
S = 3-bit operation select inputs
F = 4-bit output number
C_{N+4} = carry out of MSB position
OVR = overflow indicator

(a)

Function Table

S_2	S_1	S_0	Operation	Comments
0	0	0	CLEAR	$F_3F_2F_1F_0 = 0000$
0	0	1	B minus A	Needs $C_N = 1$
0	1	0	A minus B	Needs $C_N = 1$
0	1	1	A plus B	Needs $C_N = 0$
1	0	0	A ⊕ B	Exclusive-OR
1	0	1	A + B	OR
1	1	0	AB	AND
1	1	1	PRESET	$F_3F_2F_1F_0 = 1111$

Notes: S inputs select operation.
OVR = 1 for signed-number overflow.

(b)

FIGURE 6-15 (a) Block symbol for 74LS382/74HC382 ALU chip; (b) function table showing how select inputs (*S*) determine what operation is to be performed on *A* and *B* inputs.

Ref (1): pp 399

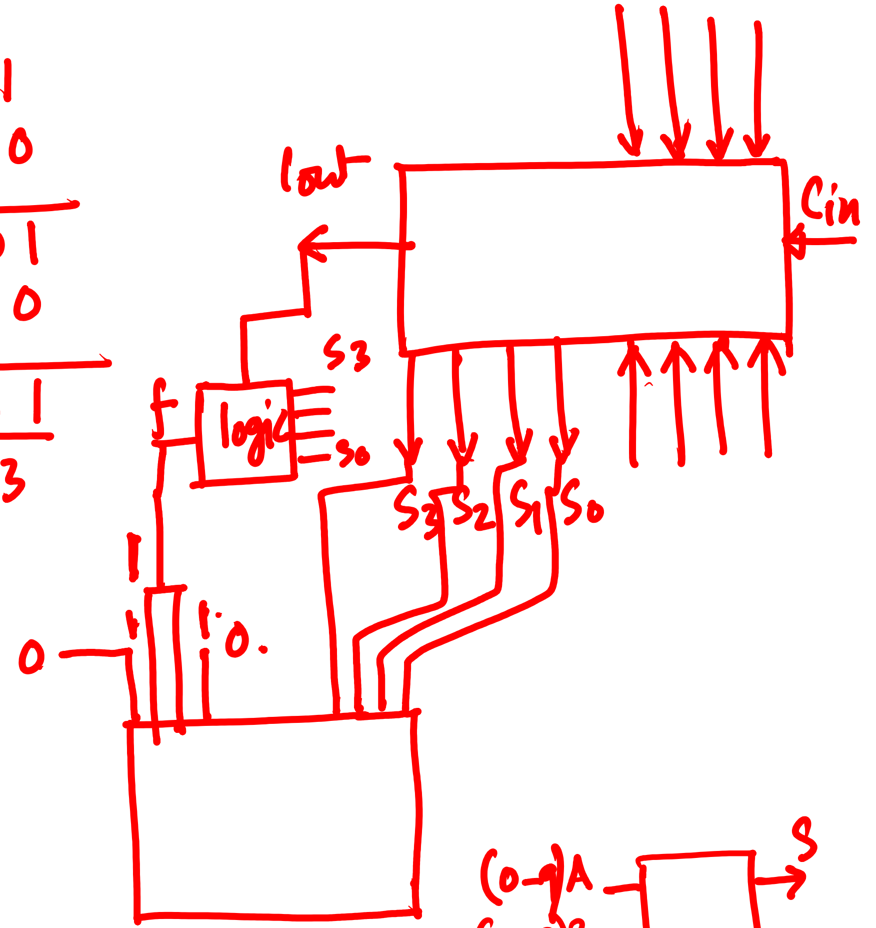
ADDER CIRCUIT

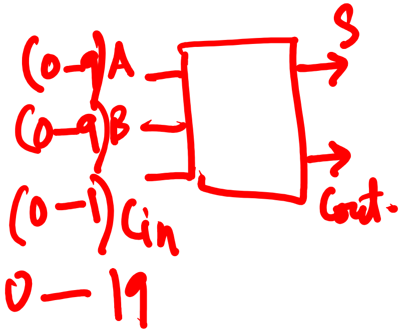
- BCD Adder:

Table 4.5
Derivation of BCD Adder

Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

$$f = K + Z_8 (Z_4 + Z_2)$$





Ref (1): pp 399

ADDER CIRCUIT

- BCD Adder:

Table 4.5
Derivation of BCD Adder

Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

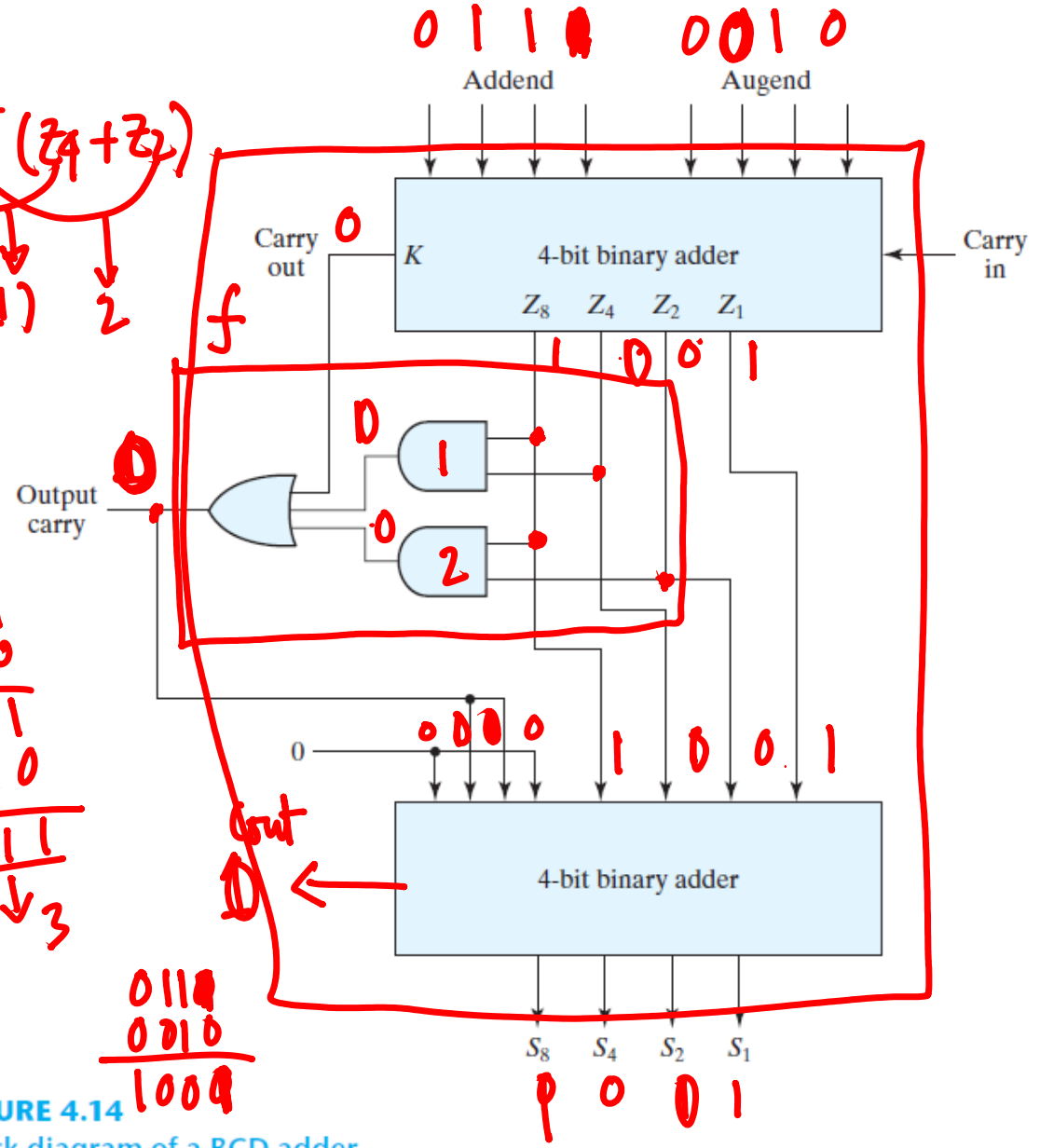


FIGURE 4.14
Block diagram of a BCD adder

Ref (2): pp 399

ADDER CIRCUIT

- Cascading BCD Adder:

783 → 0111 1000 0011 → A
+ 119 → 0001 0001 1001 → B

902 → 1000 1001 1100 ←
8 9 +0110

1000 1010 0010
+0110

← 1001 0000 0010
9 0 2