جامعة بيرزيت

BIRZEIT UNIVERSITY

ANSWER BOOKLET

Student: Digital Number 6

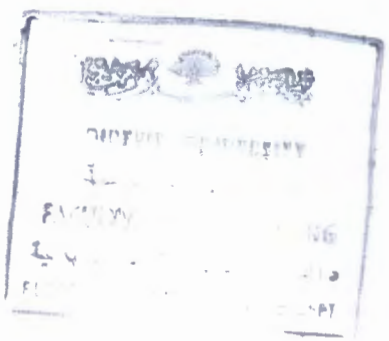
Course: Department: Number:

Division: Instructor:

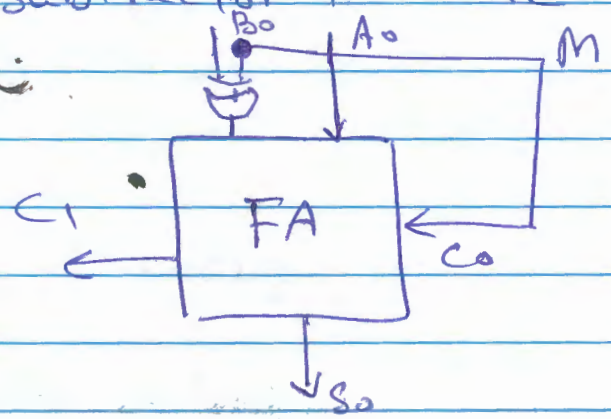
Date: Day Month Year

For Instructor's Use

Question	Grade
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
Total	



Adder - subtractor in one circuit



if $M = 0 \Rightarrow A_0 + B_0 + C_0$

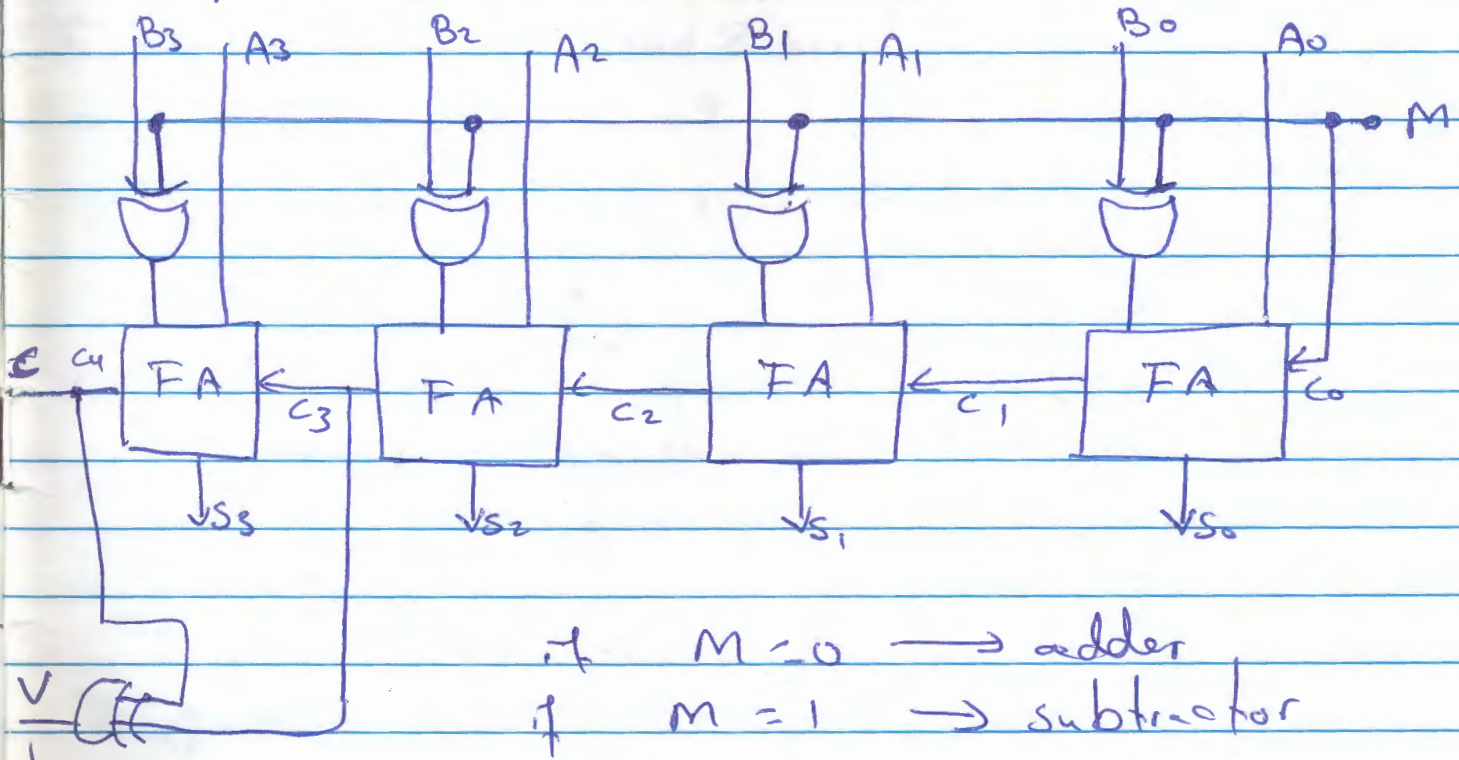
$= A_0 + B_0 + 0$ = normal addition

if $M = 1 \Rightarrow A_0 + \underbrace{B_0' + 1}$

↳ 2s complement of B_0

i.e $(A - B)$ ⇒ subtraction

⇒ four bit adder subtractor circuit



if $M = 0 \longrightarrow$ adder

if $M = 1 \longrightarrow$ subtractor

Overflow

- two numbers of n digits are added and the sum occupies n+1 digits, we say an overflow occured.
- detection of overflow is important.

- for unsigned numbers: overflow is detected by the carry after the most significant bit

- in signed numbers, if one number is +ve and the second is -ve ⇒ no overflow.

- overflow may occur if both numbers are +ve or -ve.

e.g 70 + 80 (in 8 bits register (+127 to -128))

sum = +150 (exceed 8 bits)

```
 0 1          1 0
 0 1000110    1 0111010
 0 1010000    1 0110000
---------    ---------
 1 0010110    0 1101010
```

70 + 80 → last 2 carries: 0 1

-70 - 80 → last 2 carries: 1 0

- note that the 9 bit answer will be correct (the last carry to be considered as the sign bit).
- an overflow condition can be detected by observing the carry into the sign bit position and the carry out of the sign bit position (If these are not equal), then an overflow

has occured (use XOR gate for detection)

Decimal Adder

- most arithmetic operations are in decimal.
  ⇒ numbers in binary coded form like BCD

- Decimal adder requires a minimum of nine inputs and five outputs (since four bits are required to code each decimal digit and the circuit must have an input and output carry.).

BCD Adder

- $9 + 9 + 1 = 19$ (max number for the addition of 1 digit)
  (9: first input, 9: 2nd input, 1: carry of previous step)

- Suppose we apply two BCD digits to a 4-bit binary adder (it is complex to make BCD adder because for 9 inputs we need 2^9 combinations !!).
  ⇒ The adder will form the sum in binary and produce a result that ranges from 0 through 19.

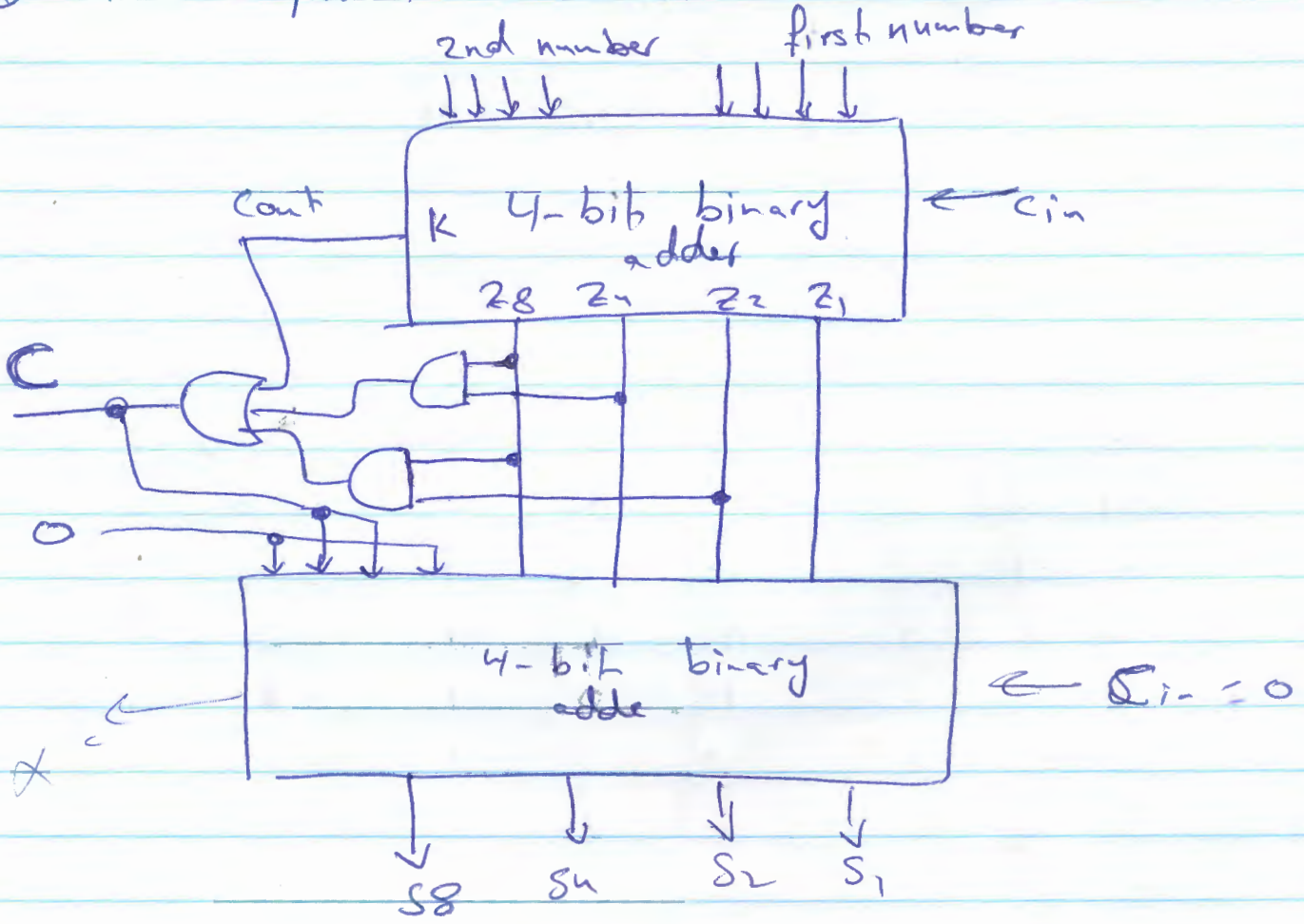
Decimal	Binary Sum					BCD Sum				
	K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0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

(*) Using K-map (with don't care conditions) we can find that

$$C = K + Z_8 Z_4 + Z_8 Z_2$$

* we know that when C = 1 in BCD addition ⇒ we must add 6 (0110 in binary) (note: for subtraction we must subtract 6).

* the final circuit



⊛ Binary Multiplier

- Example : multiplication of 2-bit numbers

A	B	result
0	0	0
0	1	0
1	0	0
1	1	1

⇒ and gate

$B_1 B_0 * A_1 A_0 = C_3 C_2 C_1 C_0$

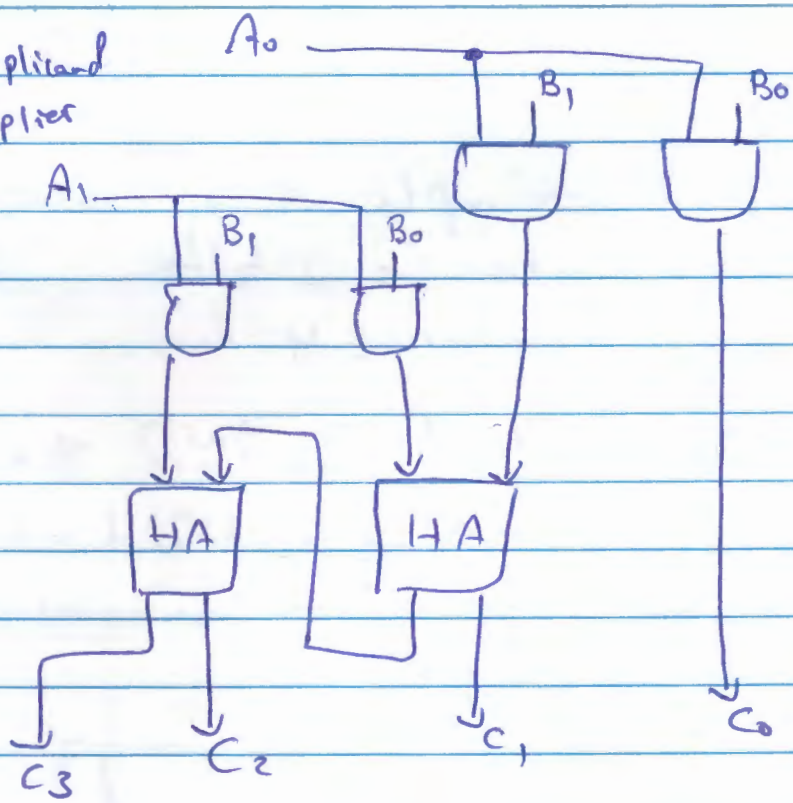
B_1 B_0 multiplicand

A_1 A_0 multiplier

A_0B_1 A_0B_0

A_1B_1 A_1B_0

C_3 C_2 C_1 C_0



- usually we need full adder instead of half adder (for more than 2-bits multiplications).

- In general, if { J multiplier / K multiplicand } we want to multiply a number of J bits with another number of K bits ⇒ we need
  (1) J × K AND gates
  (2) (J−1) K-bit adders (or (K−1) J-bits)
  (3) the product will be J+K bits

  i.e. the previous example (2-bit * 2-bit)
  ⇒ 4 and gates, 1 (2-bit adder),
  the result is 4 digits

- Another example
  - first number = 3 bits (J=3)
  - second number = 4 " (K=4)

  ⇒ we need 12 AND gates
  2 (4-bit adder)
  and we have an answer of up to 7 digits

(Figure 4-16 shows the circuit design).

Magnitude Comparator

~~3 outputs (F_1 =~~

- Compare between 2 numbers A and B
- 3 outputs $(A=B)$, $(A>B)$, $A<B$
- e.g. if A and B are 4-bit numbers
  $\Rightarrow$ 8 inputs $2^8 = 256$ combinations of inputs !!!

- use algorithmic procedure

$A = A_3 A_2 A_1 A_0$

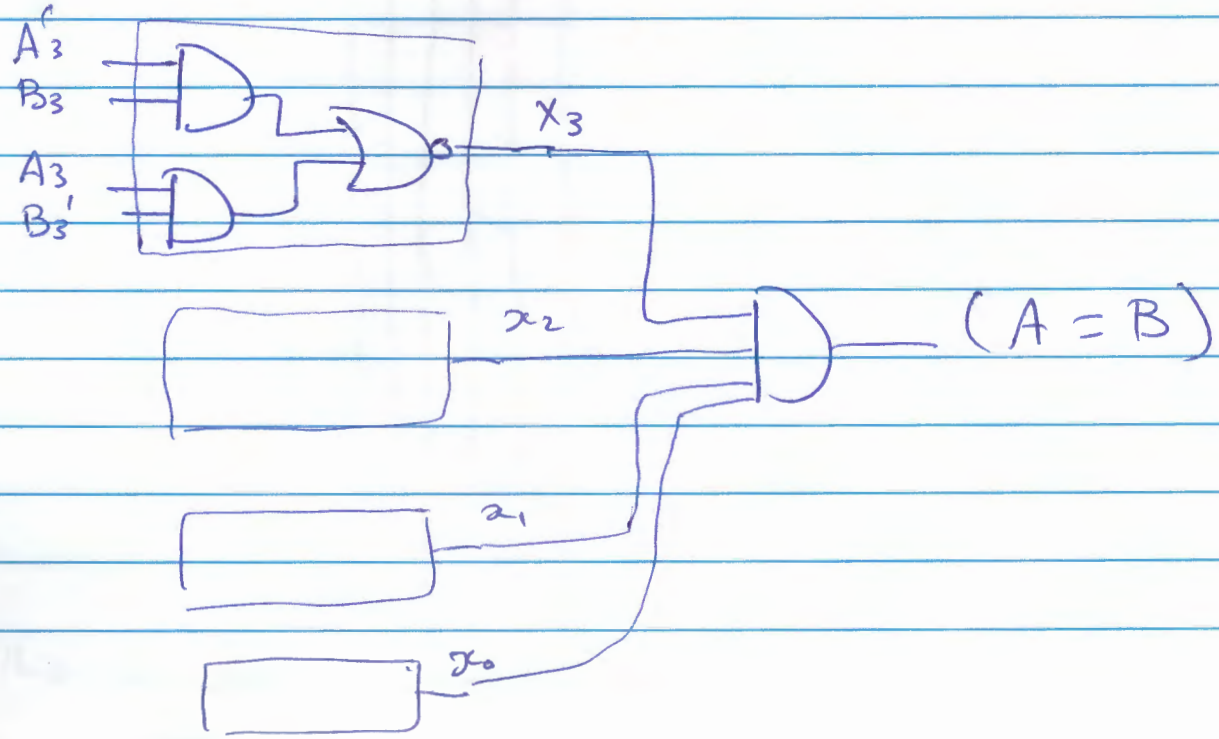
$B = B_3 B_2 B_1 B_0$

A_3	B_3	equ
0	0	1
0	1	0
1	0	0
1	1	1

XNOR $\quad x_i = A_i B_i + A_i' B_i'$

$A = B$ if all bits are equal

~~$F(A=B)$~~ $F_{A=B} = x_3 \cdot x_2 \cdot x_1 \cdot x_0$



- for A > B or B > A

$$F(A>B) = A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_1 B_1' + x_3 x_2 x_1 A_0 B_0'$$

$$F_{(A<B)} = A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1 + x_3 x_2 x_1 A_0' B_0$$

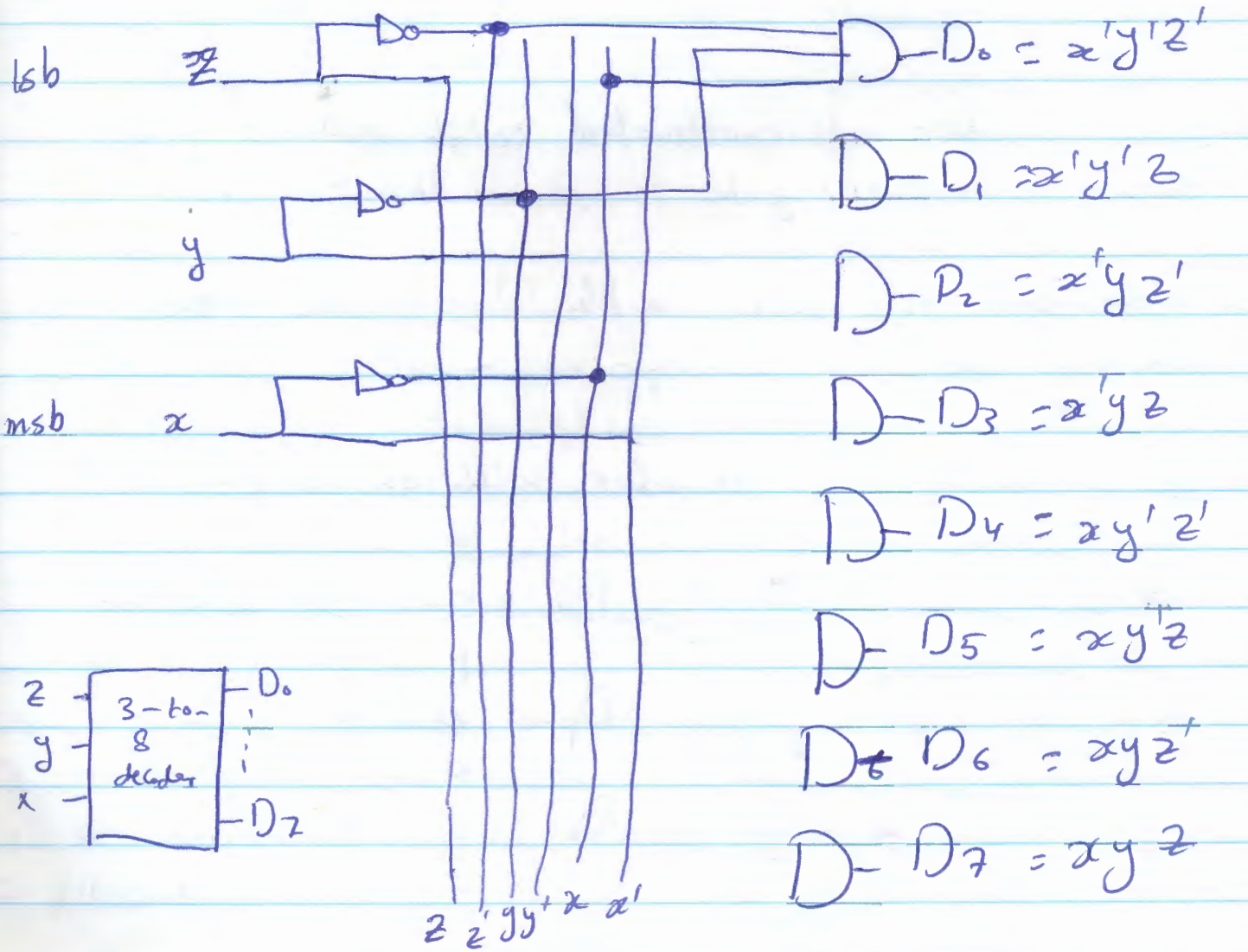
(Figure 4-17 page 134 shows the circuit).

Decoders

- A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. If the n-bit coded information has unused combinations, the decoder may have fewer than 2^n outputs.

- decoder with n lines, m lines ($m \leq 2^n$) called n-to-m decoder

e.g 3-to-8 line decoder (application: binary-to-octal conversion)



(This decoder is implemented with "and" gates)

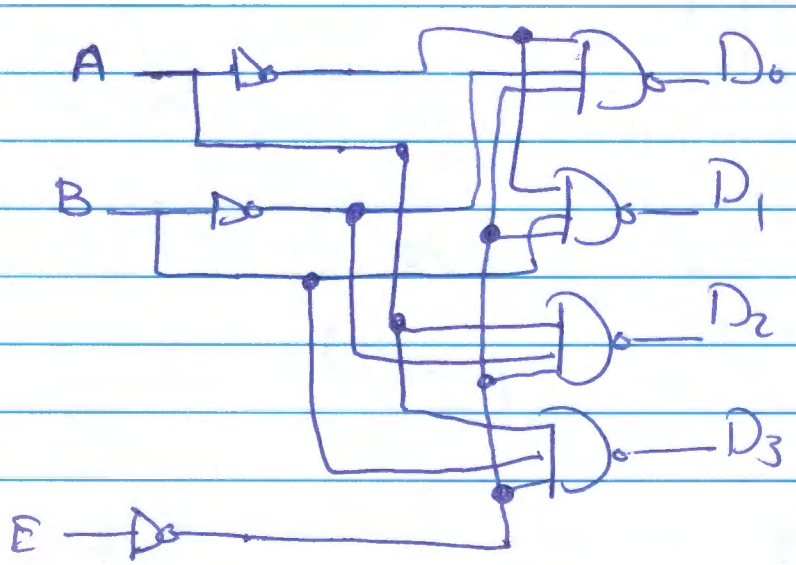
* Truth table of the decoder (3-to-8)

inputs			outputs							
x	y	z	D_0	D_1	D_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

* Some decoders are constructed with NAND gates, since a NAND gate is more commercial

- also one or more enable inputs are used to control the circuit operation

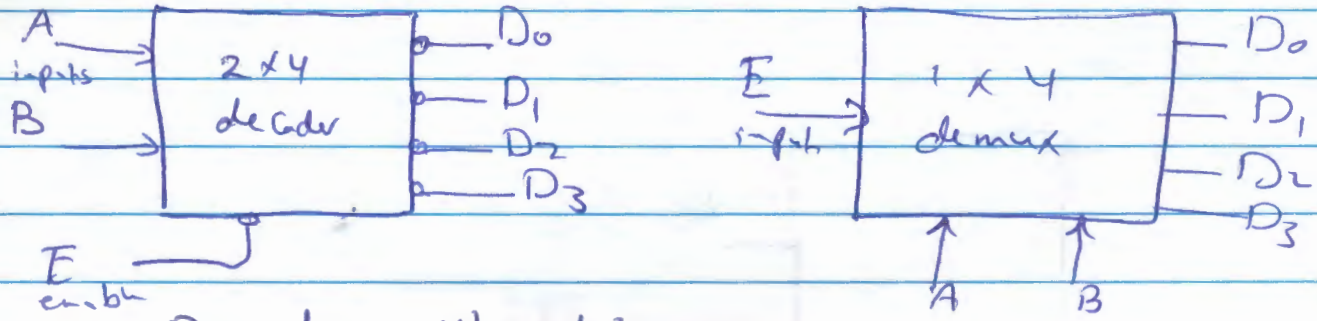
e.g 2-to-4 line decoder with an enable input E



E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

- may be more than 1 enable used to control the decoder circuit.

- A decoder with enable input can function as a demultiplexer.
- A demultiplexer is a circuit that receives information from a single line and directs it to one of 2^n possible output lines.



Decoder with enable

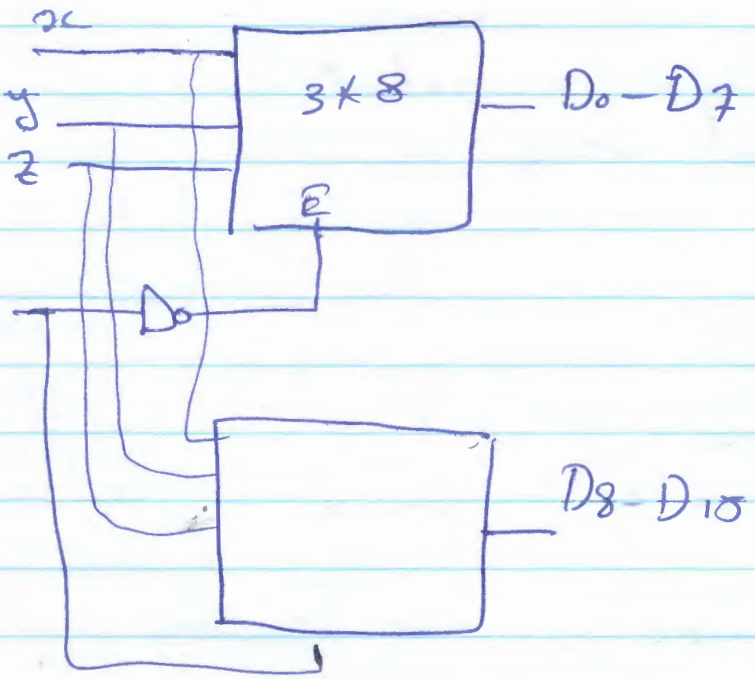
Demultiplexer

- E is taken as a data input line and A, B are taken as the selection inputs. ⟹ change from decoder to demultiplexer.
  Eg if AB = 10 ⟶ output D_2 will be same as the input E while all other outputs are maintained at 1.

- Because decoder and demultiplexer operations are obtained from the same circuit, a decoder with an enable input is referred to as a decoder/demultiplexer.

- Decoder with enable inputs can be connected together to form a larger decoder circuit

eg: 3-to-8 line decoders (2 units) ⇒ 4-to-16 line decoder



- if w=0 → top decoder is enabled, lower is disabled w=0 ⇒ 0000 - 0111

when w=1 → lower decoder is enabled

w=1 ⇒ 1000 - 1111

Combinational Logic Implementation with decoder :-

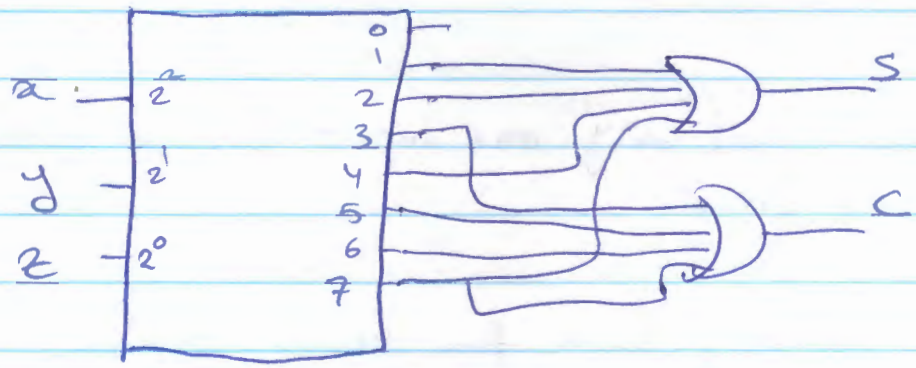
- any combinational circuit with n-inputs and m outputs can be implemented with an n-to-2^n line decoder and m OR gates

- Boolean function is expressed in sum of minterms (AND gates are internally implemented in decoder)

Example full adder

$$S(x,y,z) = \Sigma(1,2,4,7)$$
$$C(x,y,z) = \Sigma(3,5,6,7)$$

$\Rightarrow$ we need to 3-to-8 line decoder because we have 3 inputs,



- if function has lots of minterms and F' has fewer number of minterms $\Rightarrow$ implement F' and use nor gate to get f

- if decoder is implemented with nand gates $\Rightarrow$ use nand gates instead of OR.

Encoders

- inverse operation of a decoder.
- 2^n (or fewer) input lines and n output lines.
- example: octal-to-binary encoder :-

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

problem:

- it is assumed that only one input has a value of 1 at any given time.

- implementable directly from the truth table

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

⇒ the encoder can be implemented with 3 OR gates