جامعة بيرزيت

BIRZEIT UNIVERSITY

ANSWER BOOKLET

Student: Digital	Number 5
Course: Department:	Number:
Division:	Instructor:
Date: Day	Month / Year

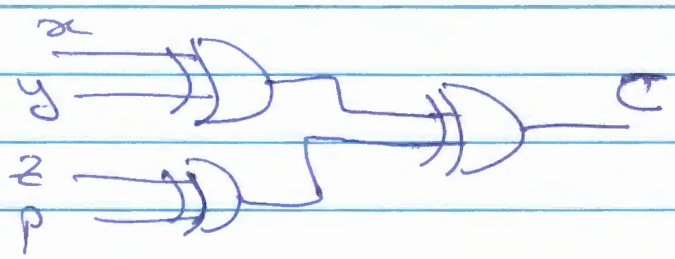
For Instructor's Use

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

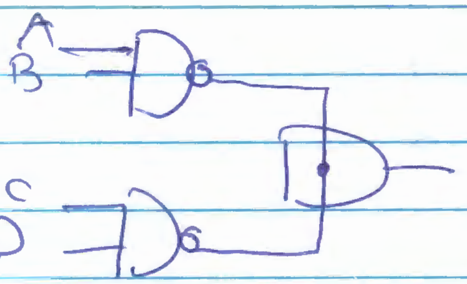
* error checker C will be used to indicate error

four bit received				Parity error check
x	y	z	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

$\Rightarrow \quad C = x \oplus y \oplus z \oplus P$

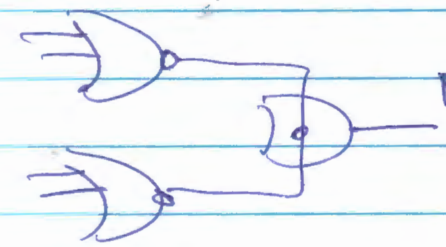


* wired logic: some nand or nor gates (but not all), allow the possibility of a wire connection between the outputs of two gates to provide a specific logic function.



$$F = (AB)' \cdot (CD)'$$
$$= (AB + CD)'$$

wired and logic



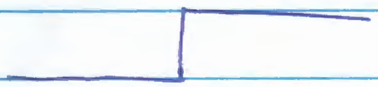
$$F = (A+B)' + (C+D)'$$
$$= [(A+B)(C+D)]'$$

wired-OR-function

from chapter 2

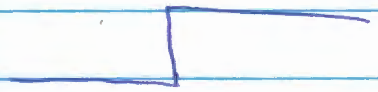
Positive and negative logic

~~Signal value~~		Logic
H		1
L		0



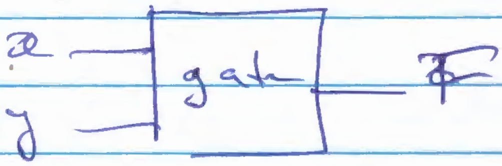
positive logic

Logic		Signal
0		H
1		L



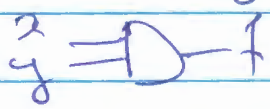
negative logic

Ex:

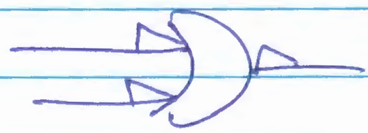


x	y	F
L	L	L
L	H	L
H	L	L
H	H	H

if positive logic ⇒ AND gate

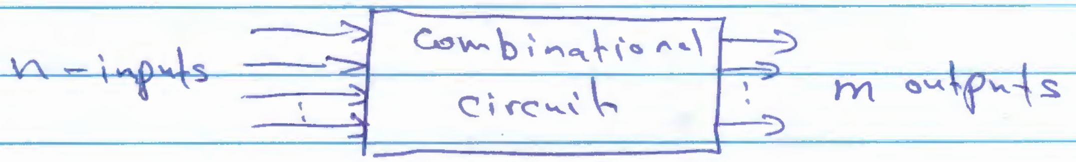


if negative logic ⇒ OR



CHAPTER 4 "Combinational Logic"

- combinational vs sequential
- combinational circuit performs an operation that can be specified logically by a set of Boolean functions.
- sequential circuits employ storage elements in addition to logic gates, and storage elements contain outputs related to previous inputs



- In combinational circuit there is no feedback paths or memory elements.

* Analysis Procedure *

Ex. Design a circuit that will convert the BCD code to an excess-3 code

Solution

- number of inputs = 4 (we use 4 digits)
  $\Rightarrow$ number of combinations $= 2^4 = 16$ inputs

let inputs are A, B, C, D

- number of outputs = 4
  let outputs are w, x, y, z

Inputs (BCD) outputs (excess-3)

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	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

AB \ CD	00	01	11	10
00				
01		1	1	1
11	X	X	X	X
10	1	1	X	X

$w = A + BC + BD$

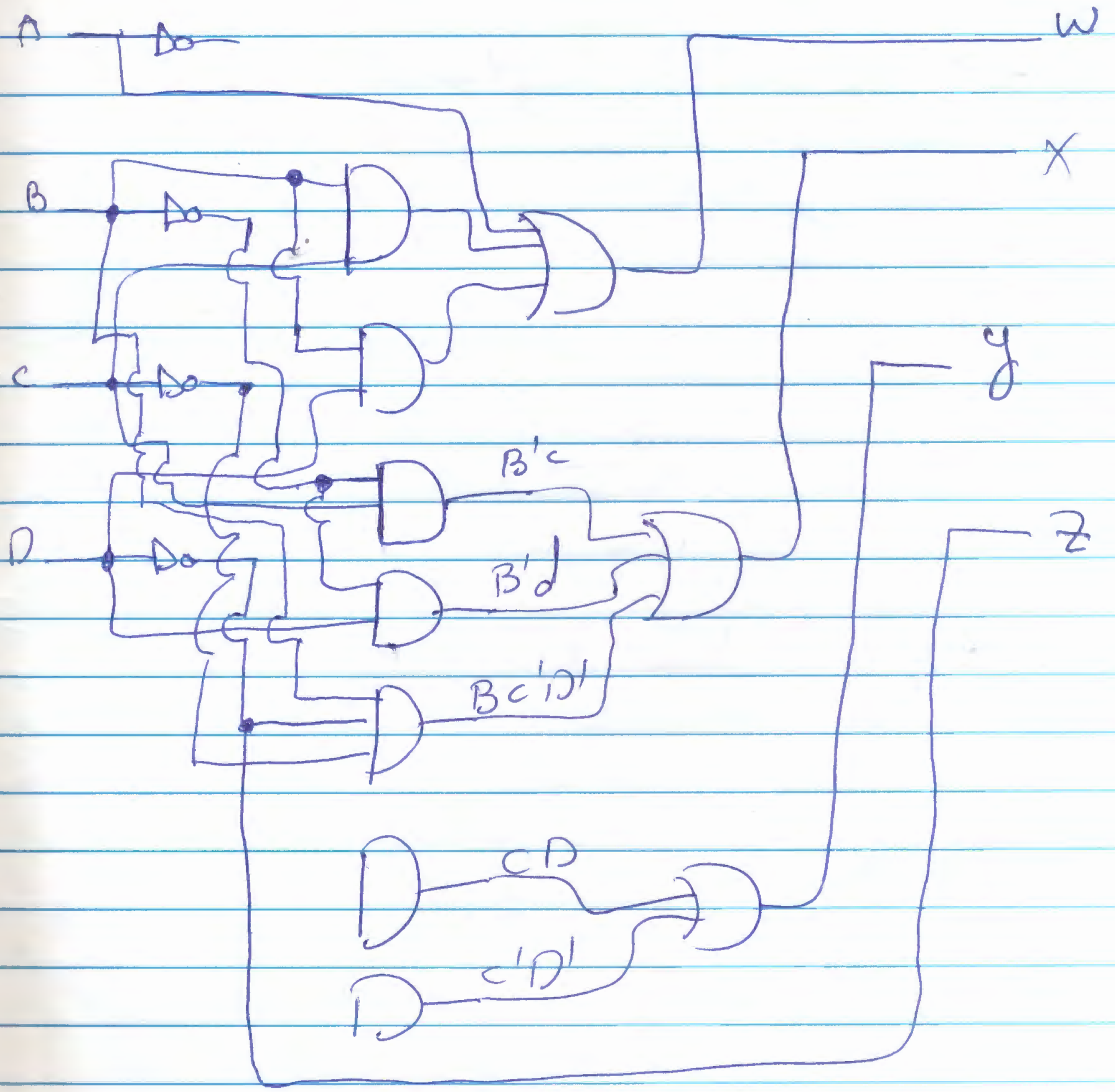
AB \ CD	00	01	11	10
00		1	1	1
01	1			
11	X	X	X	X
10		1	X	X

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

AB \ CD	00	01	11	10
00	1		1	
01	1		1	
11	X	X	X	X
10	1		X	X

AB \ CD	00	01	11	10
00	1			1
01	1			1
11	X	X	X	X
10	1		X	X

$y = CD + C'D'$ $\qquad$ $z = D'$



* note: try to manipulate the function to obtain multiple output system

e.g

$z = D'$

$y = CD + C'D' = CD + (C+D)'$

$x = B'(C+D) + B(C+D)'$

$w = A + B(C+D)$

$\Rightarrow$ easier to implement.

Binary Adder-Subtractor

- two half adders can be employed to implement a full adder

Half adder

- 2 inputs (x, y), and 2 outputs (S, C)

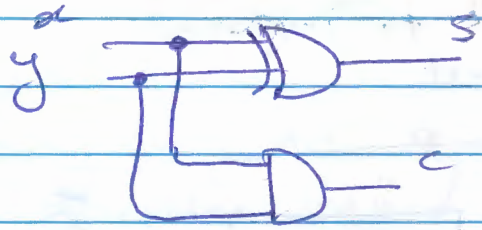
x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$\Rightarrow S = x'y + yx'$

$C = xy$

$\Rightarrow S = x \oplus y \qquad C = xy$

$\Rightarrow$



Full adder

- 3 inputs (x, y, z) and 2 outputs (S, C)

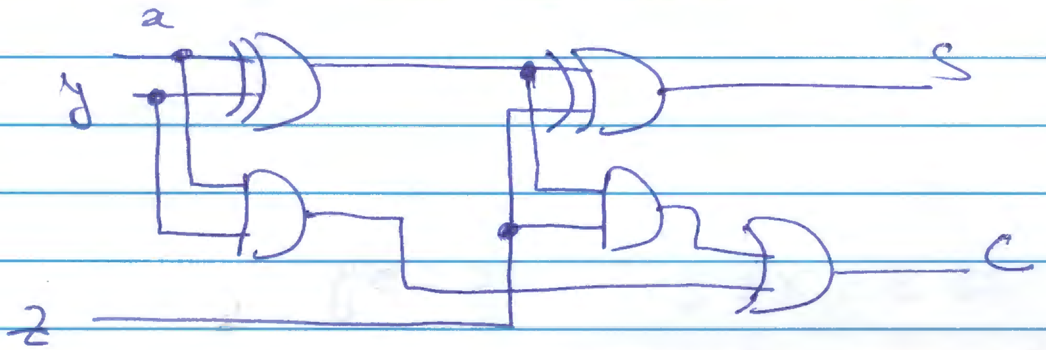
x	y	z	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
			1	1

x \ yz	00	01	11	10
0	~~1~~	1		1
1	1		1	

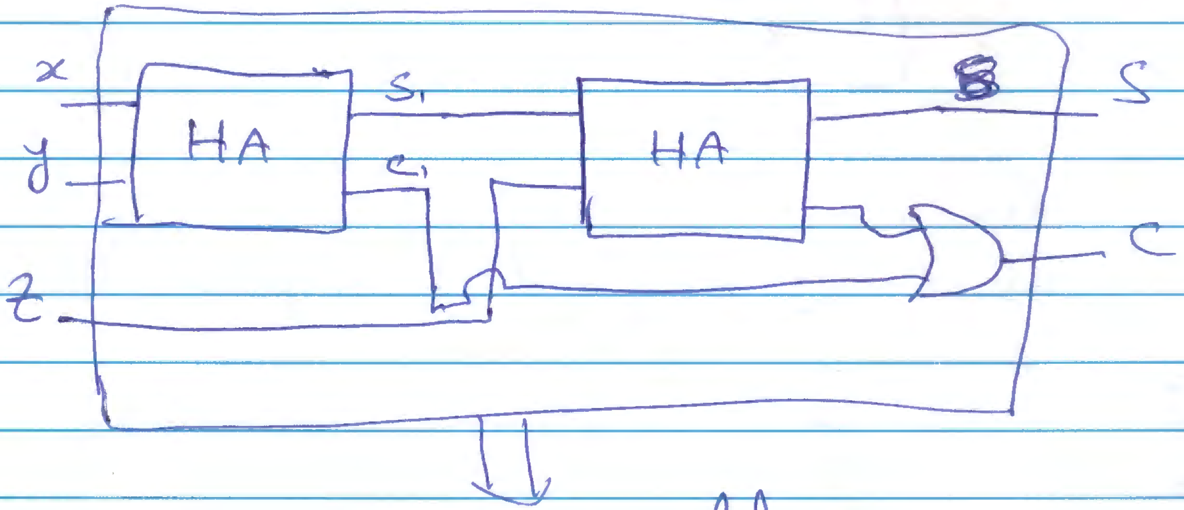
$$s = x'y'z + x'yz' + xy'z' + xyz$$
$$= x \oplus y \oplus z$$

x \ yz	00	01	11	10
0			1	
1		1	1	1

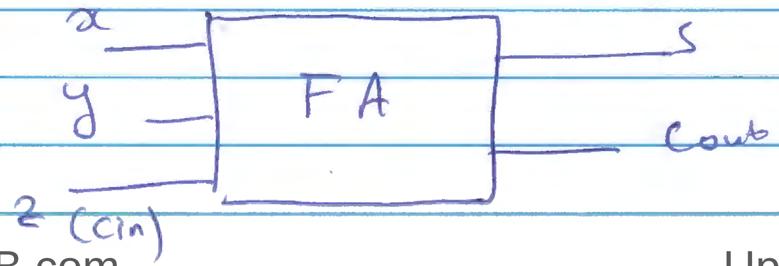
$$s = xy + xz + yz$$
$$= xy + xy'z + x'yz$$
$$= xy + z(x \oplus y)$$



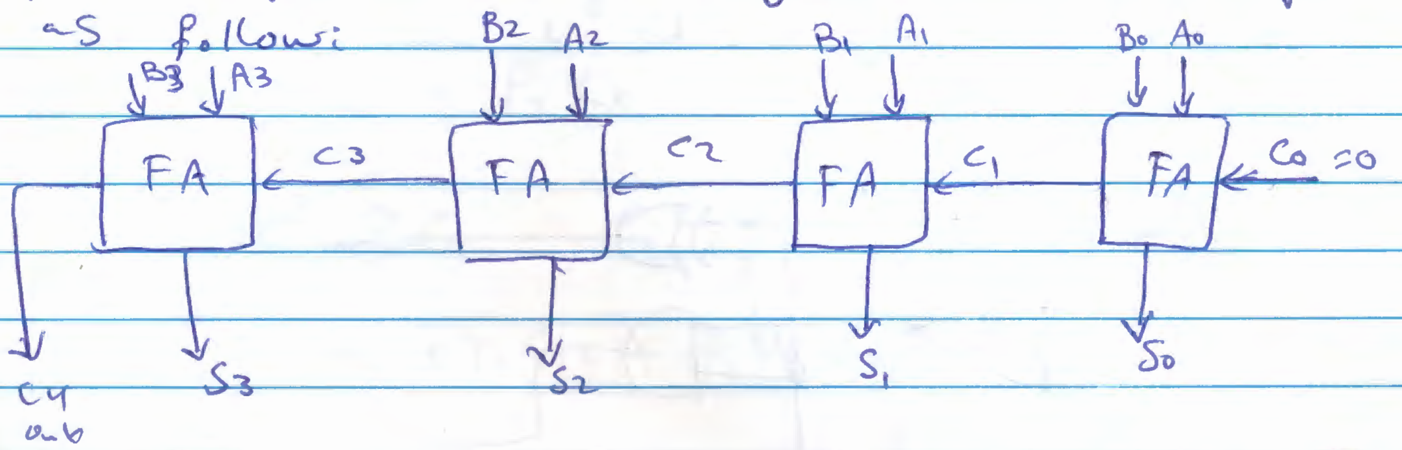
(2 half adders to implement a full adder)



full adder



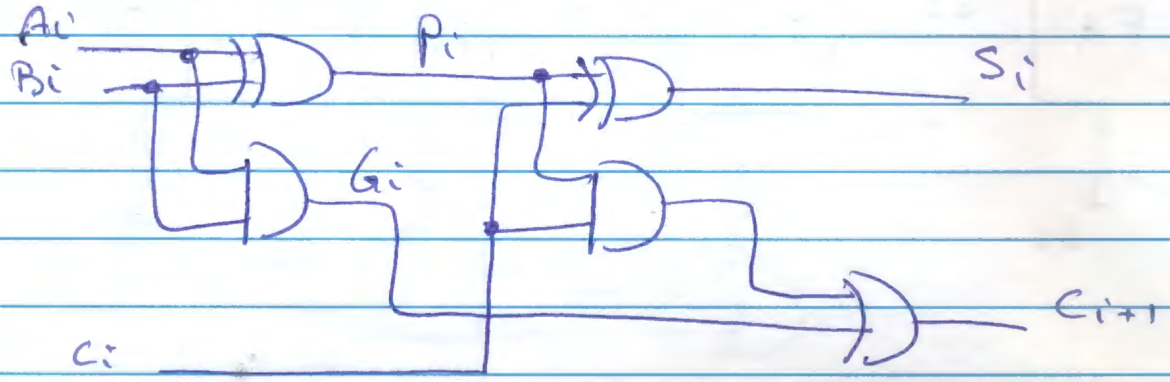
* Binary adder
- A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers.
- ~~for~~ n-bit binary adder can be implemented by using n - full adders ~~in cass~~ connected in cascade.
- for example 4-bit binary adder can be implemented as follows:



- note that the design of this circuit by the clasical method would require a truth table with $2^9 = 512$ entries, since there are 9 inputs to the circuit.
- By using an iterative method of cascading a standard function, it is possible to obtain a simple and straightforward implementation

* Carry Propagation
- time delay for each gate ⇒ propagation delay
- The longest propagation delay time in an adder is the time it takes the carry to propagate through the full adders.

- Consider S_3 in the previous circuit, inputs A_3 and B_3 are available as soon as input signals are applied to the adder. However, input carry C_3 doesn't settle to its final value until C_2 is available from the previous stage. Similarly, C_2 has to wait for C_1 and so on.



- P_i and G_i are common to all full adders and depend only on the values of A and B.

- C_i to propagate to C_{i+1} need to propagate by 2 gate levels ⇒ for C_0 to propagate to C_4, there are $2 \times 4 = 8$ gate level, for n bit adder → $2n$ gate level.

* Carry lookahead

$P_i = A_i \oplus B_i$ (Carry Propagate)

$G_i = A_i \cdot B_i$ (carry generate)

* $S_i = P_i \oplus C_i$

$C_{i+1} = G_i + P_i C_i$

- Now write the boolean functions for the carry outputs of each stage and substitute for each C_i its value from the previous equations:

C_0 = input carry

$C_1 = G_0 + P_0 C_0$

$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$

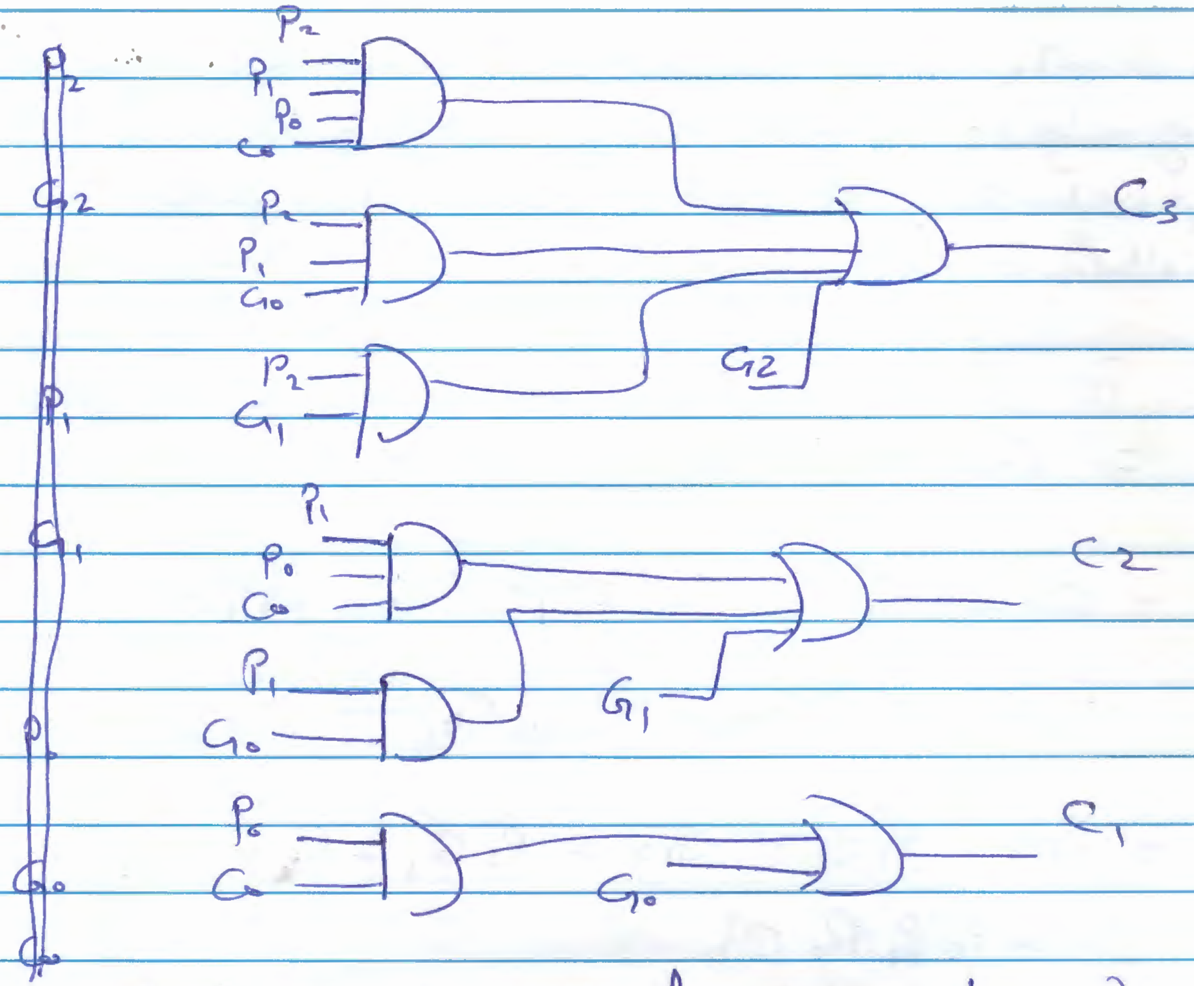
$= G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0$

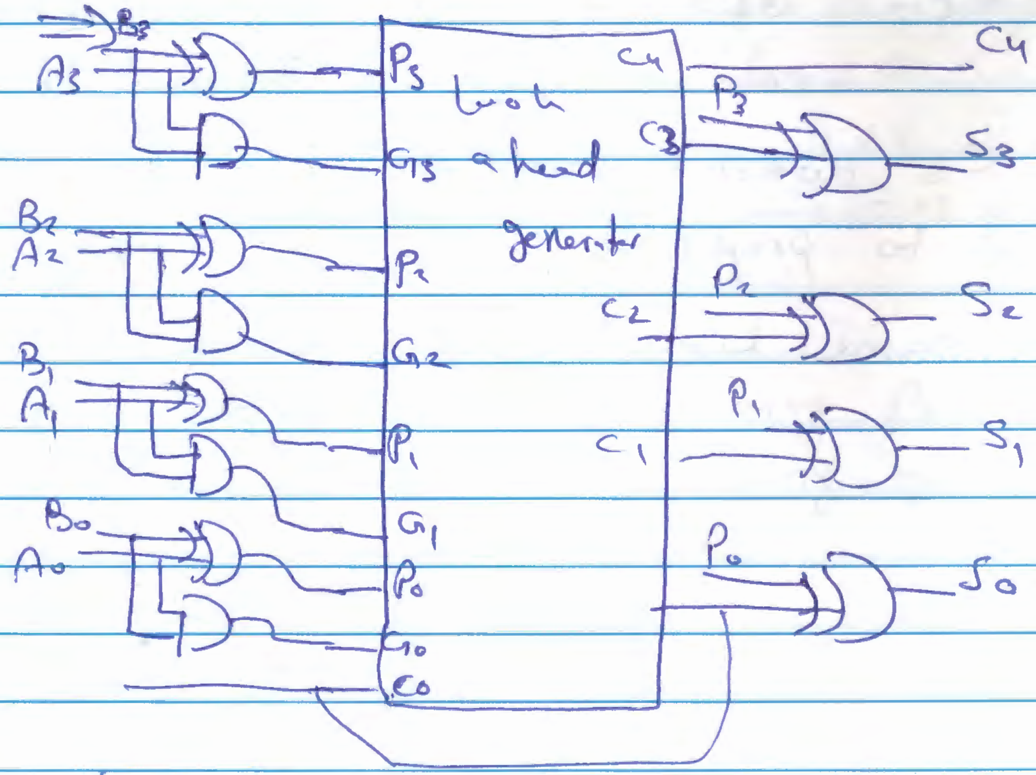
$+ P_2 P_1 P_0 C_0$

- now we expressed C_1, C_2, and C_3 in term of G, P, and C_0 ⟹ that means in term of inputs A, B, and C_0.

⟹ C_3 doesn't have to wait for C_2 and C_1 to propagate, C_3 is propagated at the same time as C_1 and C_2.



(Carry Lookahead Generator).



(4-bit Adder with carry lookahead).

① Binary subtractor

② $A - B = A + 2s$ complement of B

- 2s complement of B is 1s complement of B + 1

$\Rightarrow$ A − B becomes A plus 1s complement of B, plus 1.

- for unsigned number the result is A − B if $A \geq B$ or the 2s complement of B − A if $A < B$.

~~A + 2s complement of B~~

eg 111 − 101

1 1 1
1 0 1
0 1 0 (A − B, A > B)

b a 101 − 111

1 0 1
0 0 1
0 0 1
1 1 0

2s comp 0 0 1 + 1 = 010 ✓

e.g. $A = 0\ 1\ 1\ 1$ $B = 0\ 1\ 0\ 1$

$\Rightarrow A - B = A + 2s$ comp of B

$$
\begin{array}{rcccc}
\Rightarrow & 0 & 1 & 1 & 1 \\
+ & 1 & 0 & 1 & 1 \\
\hline
\cancel{1} & 0 & 0 & 1 & 0
\end{array}
$$

$(A-B)$ ✓

if $A = 0\ 1\ 0\ 1$ $B = 0\ 1\ 1\ 1$

$\Rightarrow A - B = A + 2s$ complement of B

$$
\begin{array}{cccc}
0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 \\
\hline
1 & 1 & 1 & 0
\end{array}
$$

$\Rightarrow 2s$ comp $= 0\ 0\ 1\ 0$ (✓)

- for signed numbers, the result is $A-B$, provided that there is no overflow.