Digital Logic Circuits

Sources

Lecture Notes

Introduction to Electronics, D Cory, J Freidberg, MIT
Digital Logic Part 1, Part 2, Part 3
http://www.mit.edu:8001/courses/6.071/lectures/L21.pdf
http://www.mit.edu:8001/courses/6.071/lectures/L22.pdf
http://www.mit.edu:8001/courses/6.071/lectures/L23.pdf

Technical Information Sciences 1, B. Blatter, Swiss Federal Institute of Technology

Books

Building Scientific Apparatus, John H Moore, Christopher C. Davis, Michael A Coplan, Perseus Books.

WEB

www.play-hookey.com
http://www.ee.surrey.ac.uk/Projects/Labview/gatesfunc/

Boolean Postulates

$$0 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

$$1 \cdot 0 = 0$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$

$$0 + 0 = 0$$

$$1 + 0 = 1$$

$$\bar{1} = 0$$

$$\bar{0} = 1$$

Laws of Boolean Algebra

Communicative

$$A + B = B + A$$

$$A \cdot B = B \cdot A$$

Associative

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

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

Distributive

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

$$(A + B) \cdot (C + D) = A \cdot B + A \cdot D + B \cdot C + B \cdot D$$

de Morgan

$$\overline{A \cdot B \cdot C} = \overline{A} + \overline{B} + \overline{C}$$

$$\overline{A + B + C} = \overline{A} \cdot \overline{B} \cdot \overline{C}$$

Identities

$$A + A = A$$

$$A \cdot A = A$$

$$AB + A\overline{B} = A$$

$$(A + B) \cdot (A + \overline{B}) = A$$

Redundance

$$A + A \cdot B = A$$

$$A + \overline{A} \cdot B = A + B$$

$$\overline{A} + A \cdot B = \overline{A} + B$$

$$A \cdot (A + B) = A$$

$$A \cdot (\overline{A} + B) = A \cdot B$$

$$0 + A = A$$

$$0 \cdot A = 0$$

$$1 + A = 1$$

$$1 \cdot A = A$$

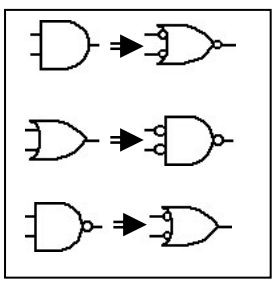
$$\overline{A} + A = 1$$

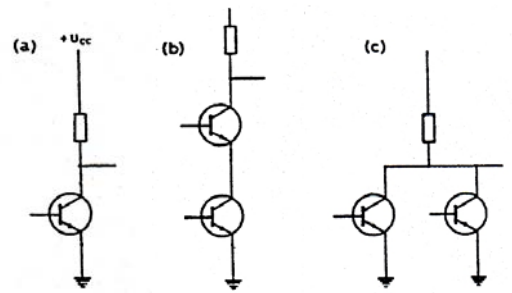
$$\overline{A} \cdot A = 0$$

Logic Gates & Boolean Expression

OR		$A + B$		
AND	coincide	$A \cdot B$		
Buffer		A		
NOT	invert	$\overline{A}$		
NOR	not of quantity A or B	$\overline{A + B}$,	$\overline{A} \cdot \overline{B}$ (not A and not B)	
NAND		$\overline{A \cdot B}$,	$\overline{A} + \overline{B}$	
XOR	anti coincide	$A \oplus B$,	$(\overline{A} \cdot B) + (A \cdot \overline{B})$,	$(A + B) \cdot \overline{A \cdot B}$
NXOR	equivalence	$\overline{A \oplus B}$,	$\overline{A} \cdot \overline{B} + A \cdot B$	

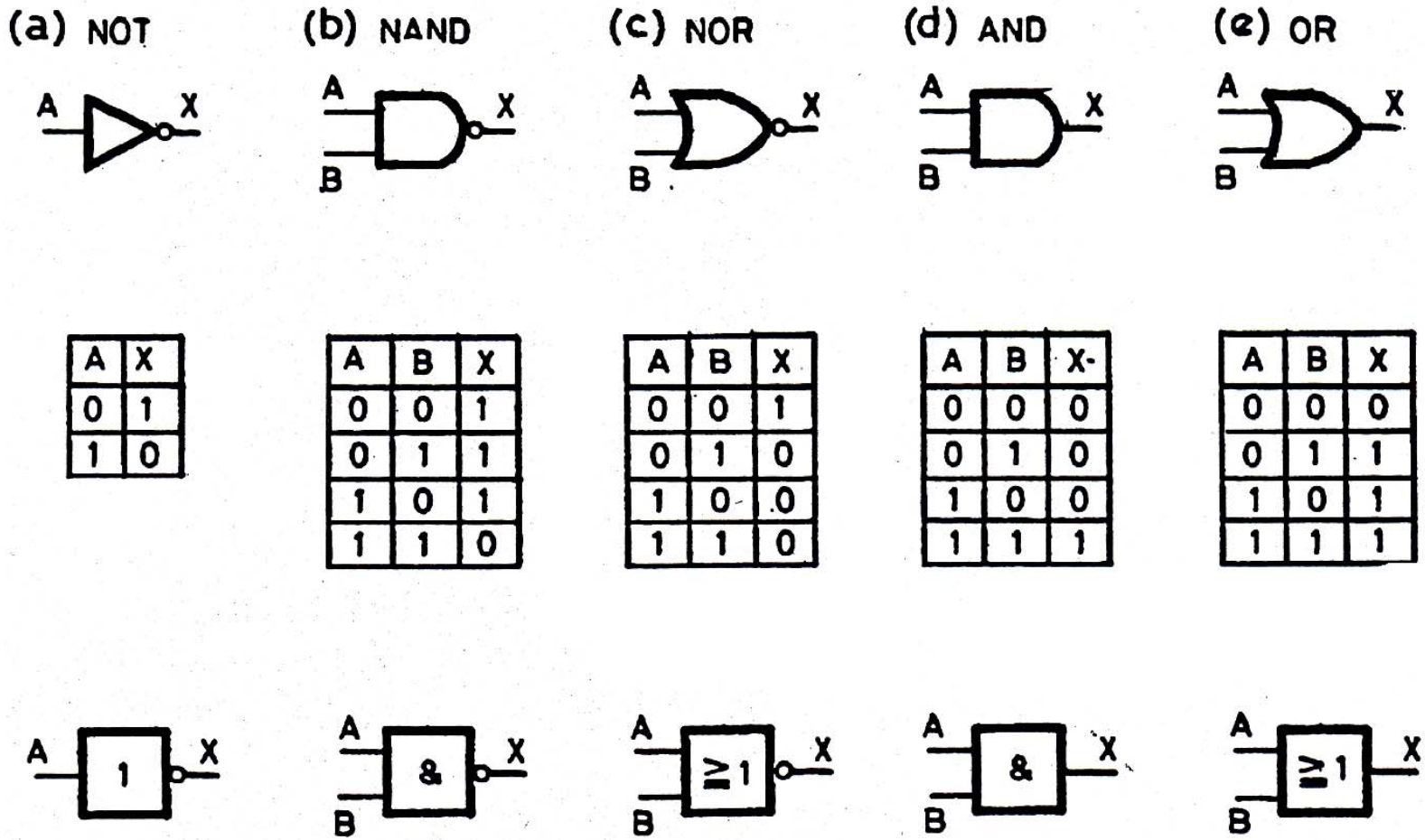
NAND and NOR are universal functions and can be used to construct any of the others.





NOT **NAND** **NOR**

Logic Gates & Boolean Expression



A	X
0	1
1	0

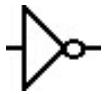
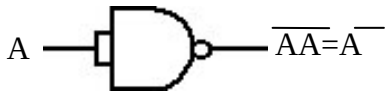
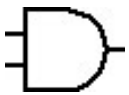
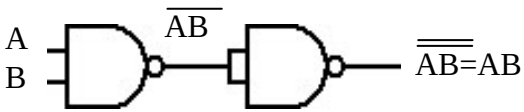
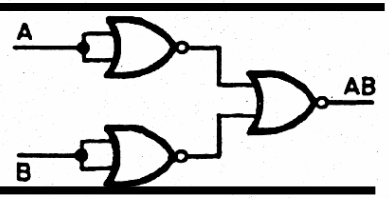
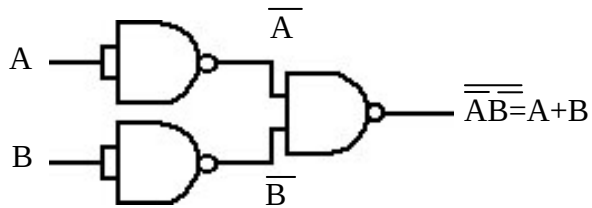
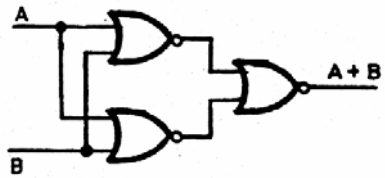
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

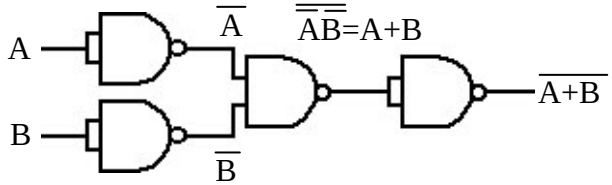
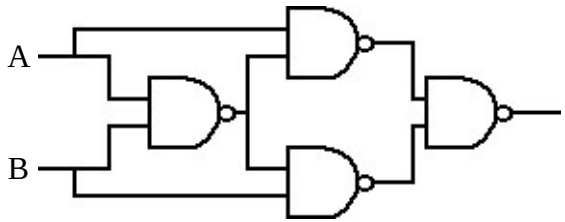
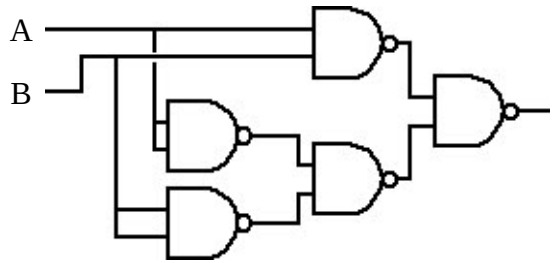
A	B	X-
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Equivalents (1)

Logic Gate	NAND Equivalent	NOR Equivalent
NOT 		
AND 		
NAND 		
OR 		

Equivalents (2)

Logic Gate	NAND Equivalent	NOR Equivalent
NOR 		
XOR 		
XNOR 		

Binary Adder

Adding: Adding binary numbers is just like adding decimal numbers; whenever the result of adding one column of numbers is greater than one digit, a 1 is carried over to the next column to be added.

Example:

			1		1			
20_{10} =	0	0	0	1	0	1	0	0
87_{10} =	0	1	0	1	0	1	1	1
	0	1	1	0	1	0	1	1

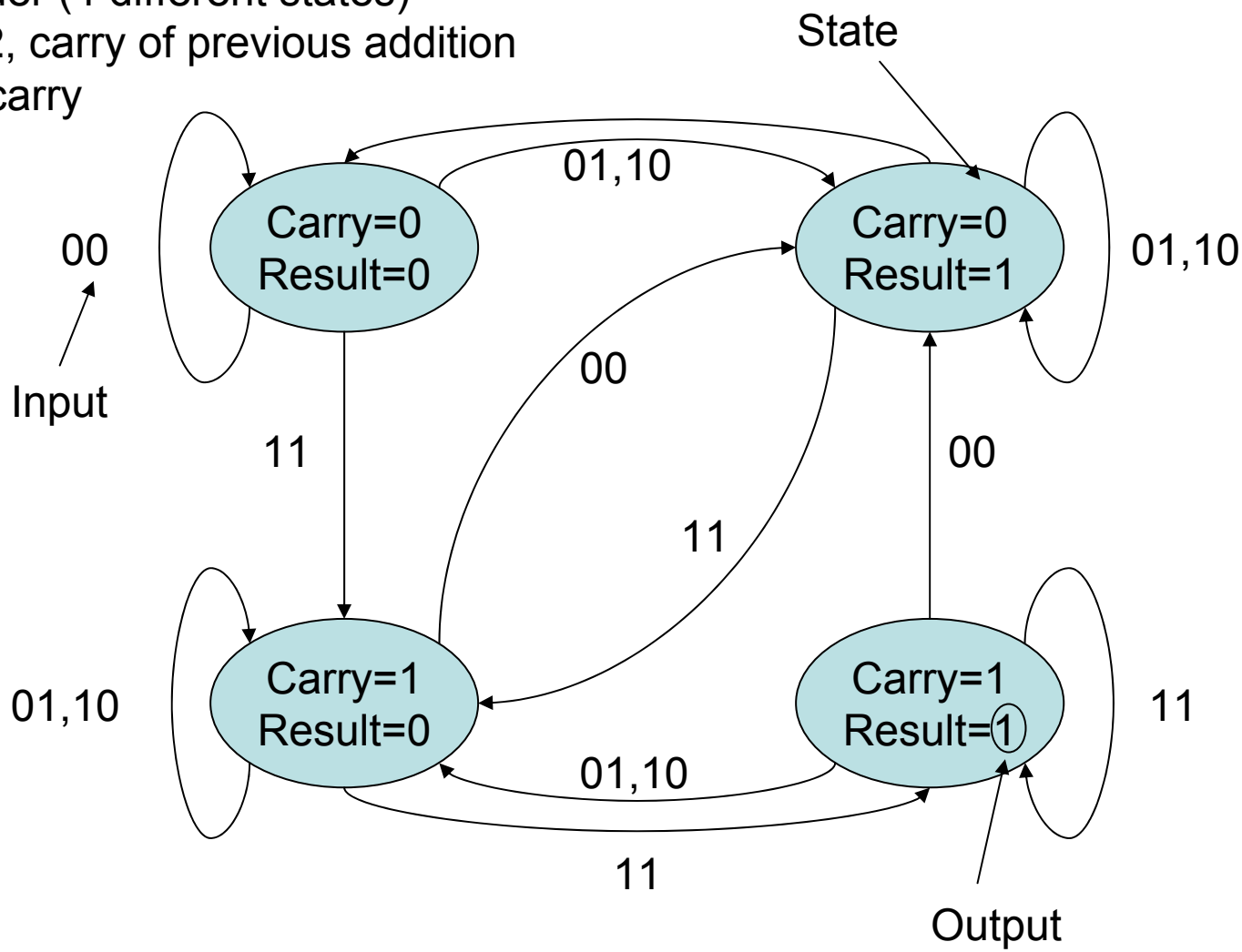
Subtraction: This operation is similar to decimal subtraction. The only potential point of confusion is that when "borrowing" from one column a 10_2 (=2_{10}) is carried over (not just a 1) to the next column. A trick to subtracting binary numbers is to add the 2's compliment of the subtracting number to the true binary representation of the number from which it is subtracting.

Example:

$+19_{10}$ =	0	0	0	1	0	0	1	1
-7_{10} =	1	1	1	1	1	0	0	1
Sum =	0	0	0	0	1	1	0	0

States of Binary Adder

Binary Adder (4 different states)
input x1,x2, carry of previous addition
output: y, carry



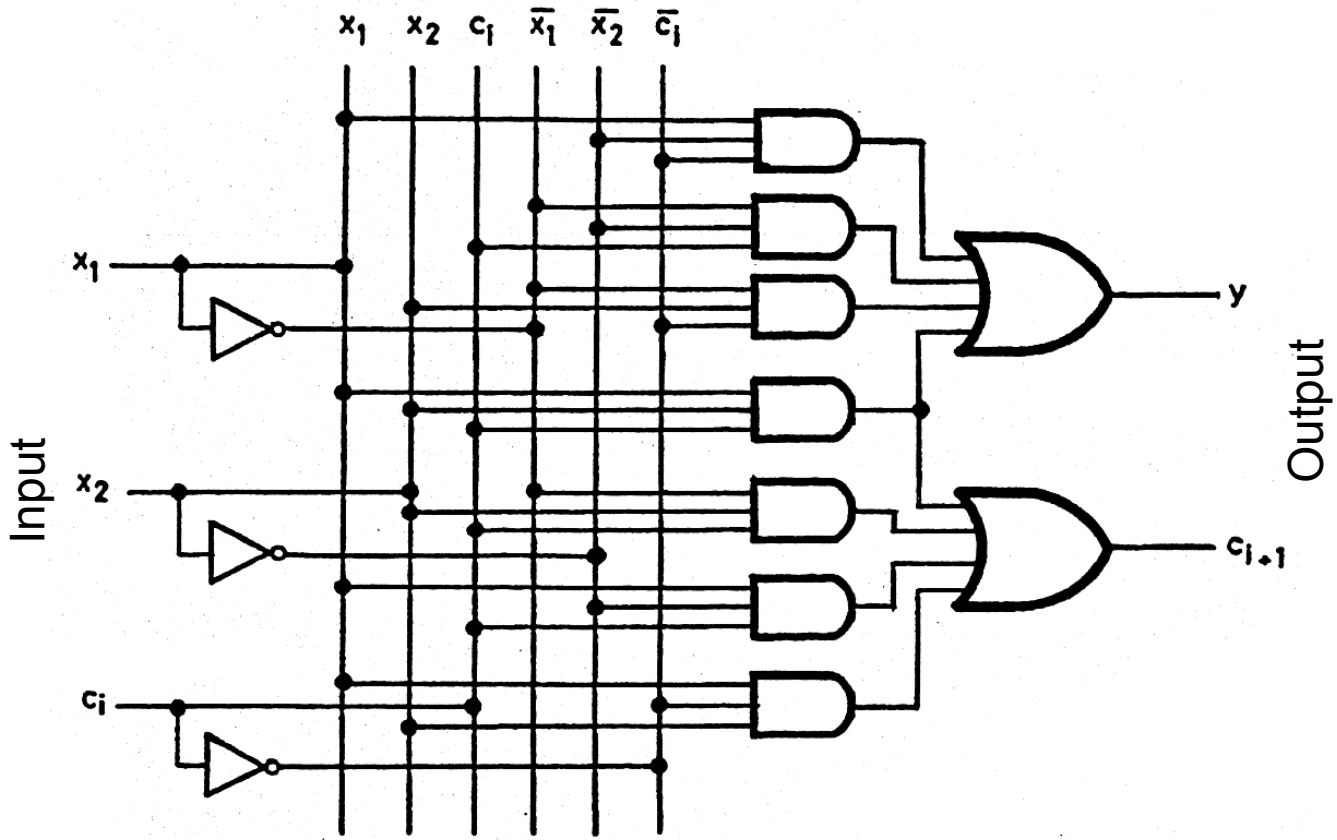
Logical Expression

Input	Output	Min		Max	
$x_1\ x_2\ c_i$	$y\ c_{i+1}$	y	c_{i+1}	y	c_{i+1}
0 0 0	0 0			$x_1+x_2+c_i$	$x_1+x_2+c_i$
0 0 1	1 0	$\overline{x}_1\overline{x}_2c_i$			$x_1+x_2+\overline{c}_i$
0 1 0	1 0	$\overline{x}_1x_2\overline{c}_i$			$x_1+\overline{x}_2+c_i$
0 1 1	0 1		$\overline{x}_1x_2c_i$	$x_1+\overline{x}_2+\overline{c}_i$	
1 0 0	1 0	$x_1\overline{x}_2\overline{c}_i$			$\overline{x}_1+x_2+c_i$
1 0 1	0 1		$x_1\overline{x}_2c_i$	$\overline{x}_1+x_2+\overline{c}_i$	
1 1 0	0 1		$x_1x_2\overline{c}_i$	$\overline{x}_1+\overline{x}_2+c_i$	
1 1 1	1 1	$x_1x_2c_i$	$x_1x_2c_i$		

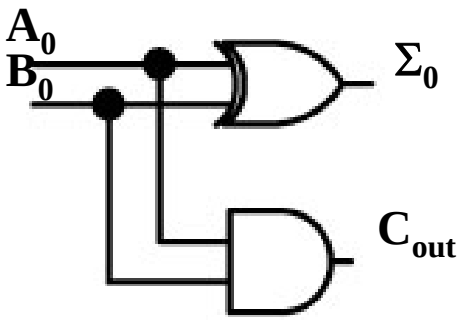
$$y = \overline{x}_1\overline{x}_2c_i + \overline{x}_1x_2\overline{c}_i + x_1\overline{x}_2\overline{c}_i + x_1x_2c_i$$

$$c_{i+1} = \overline{x}_1x_2c_i + x_1\overline{x}_2c_i + x_1x_2\overline{c}_i + x_1x_2c_i$$

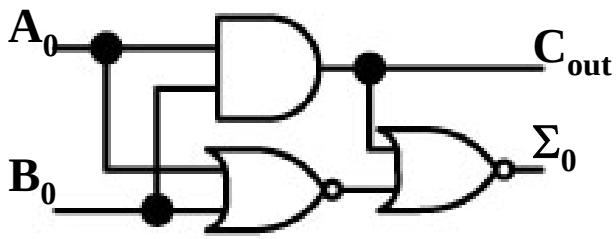
Circuit



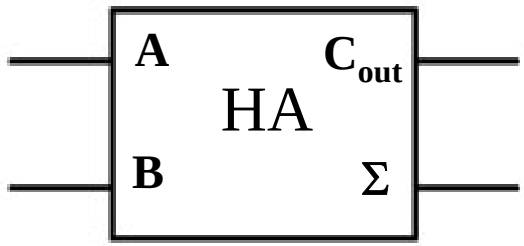
Half Adder



Alternative Representation



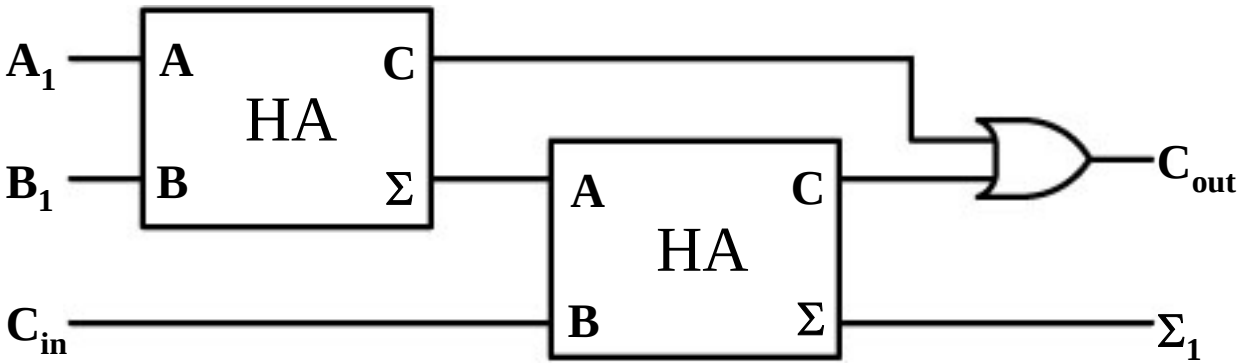
Half-adder symbol



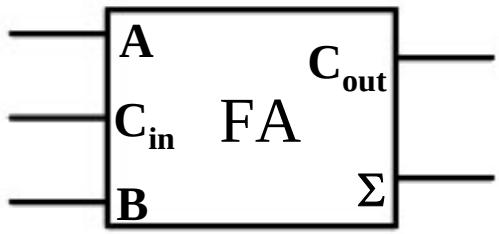
These circuits add two one-bit numbers and produce one two-bit number with a carry.

Full Adder

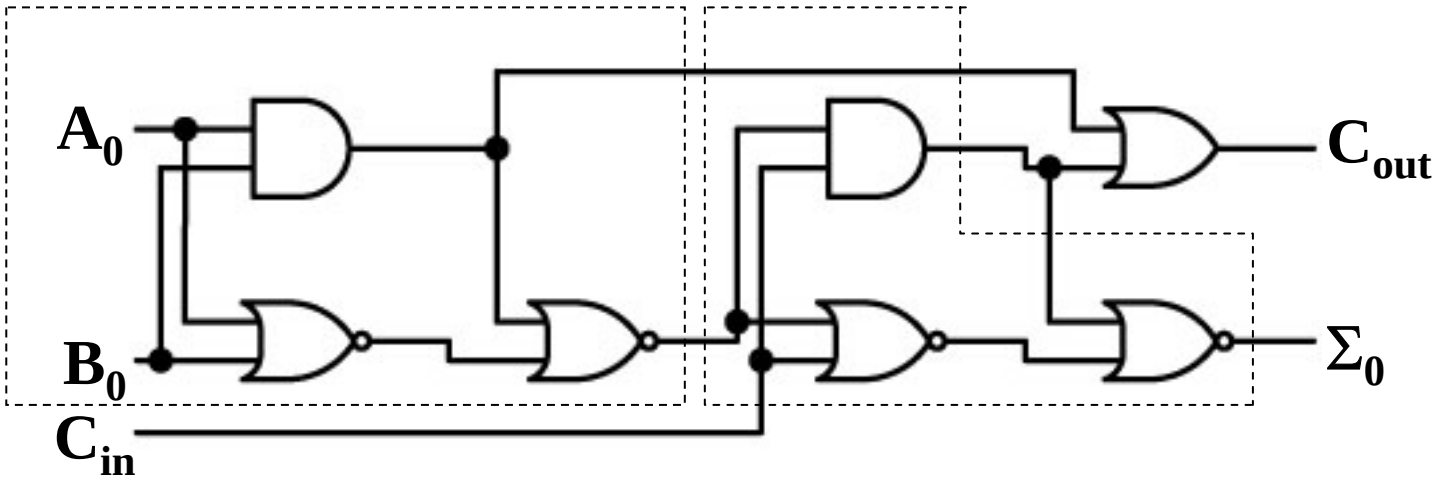
Full-Adder:



full-adder symbol



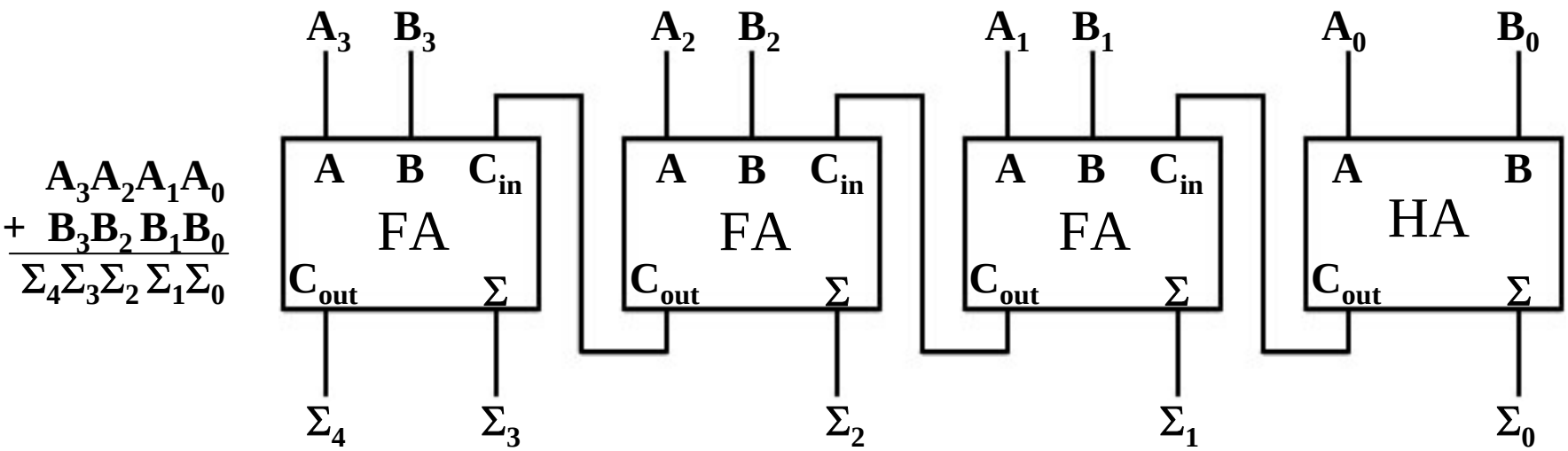
Full Adder Implementation



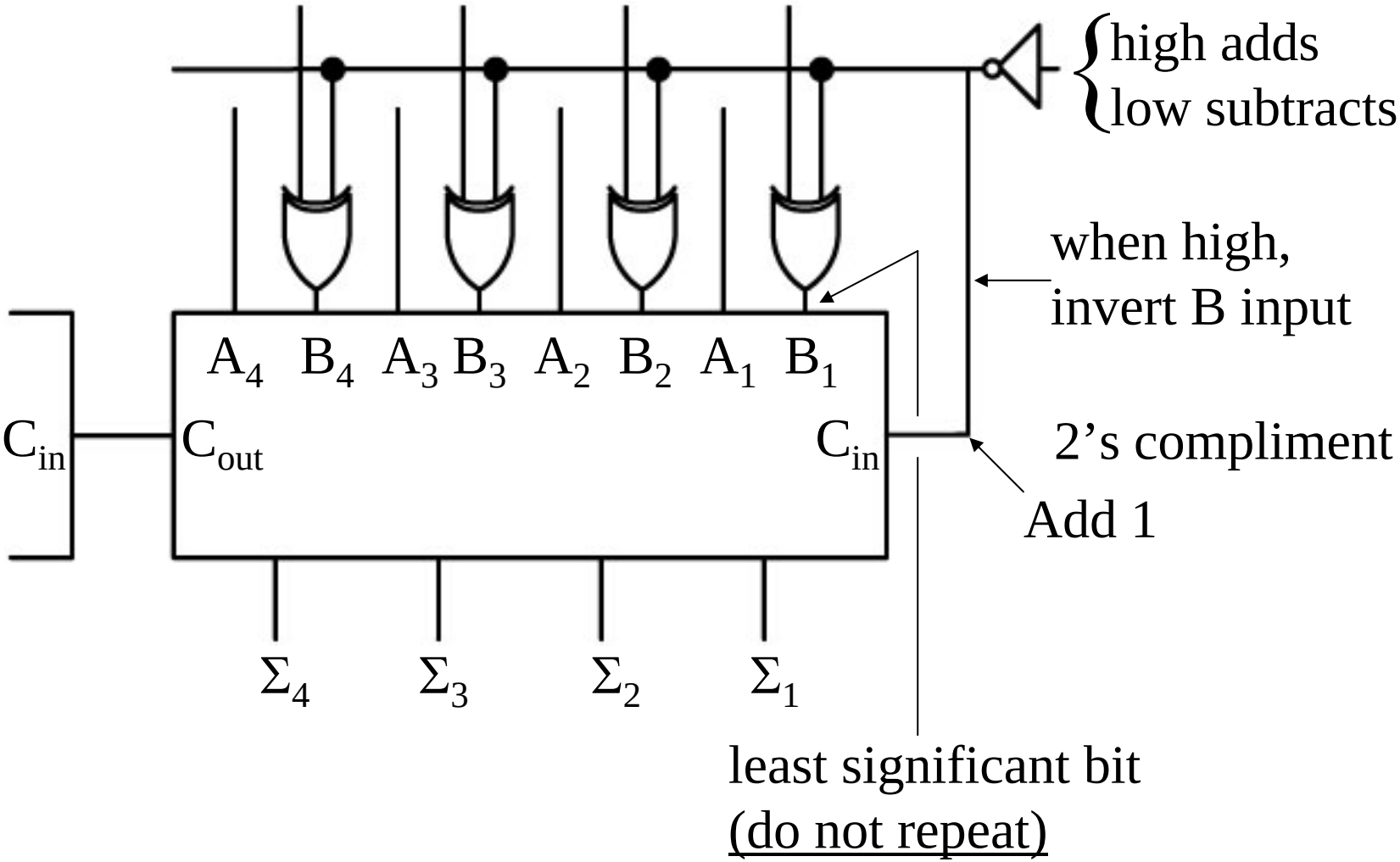
4bit Adder

Now, with the half-adders and full-adders, a multi-bit adder can be constructed:

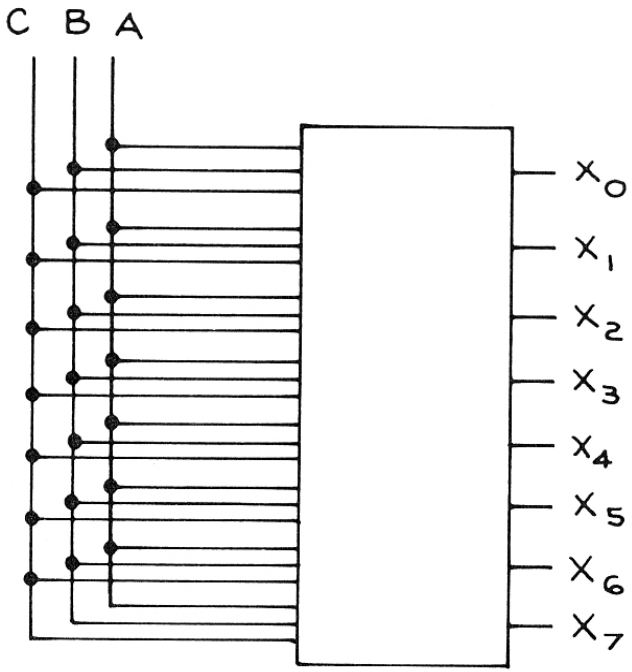
4-bit Adder:

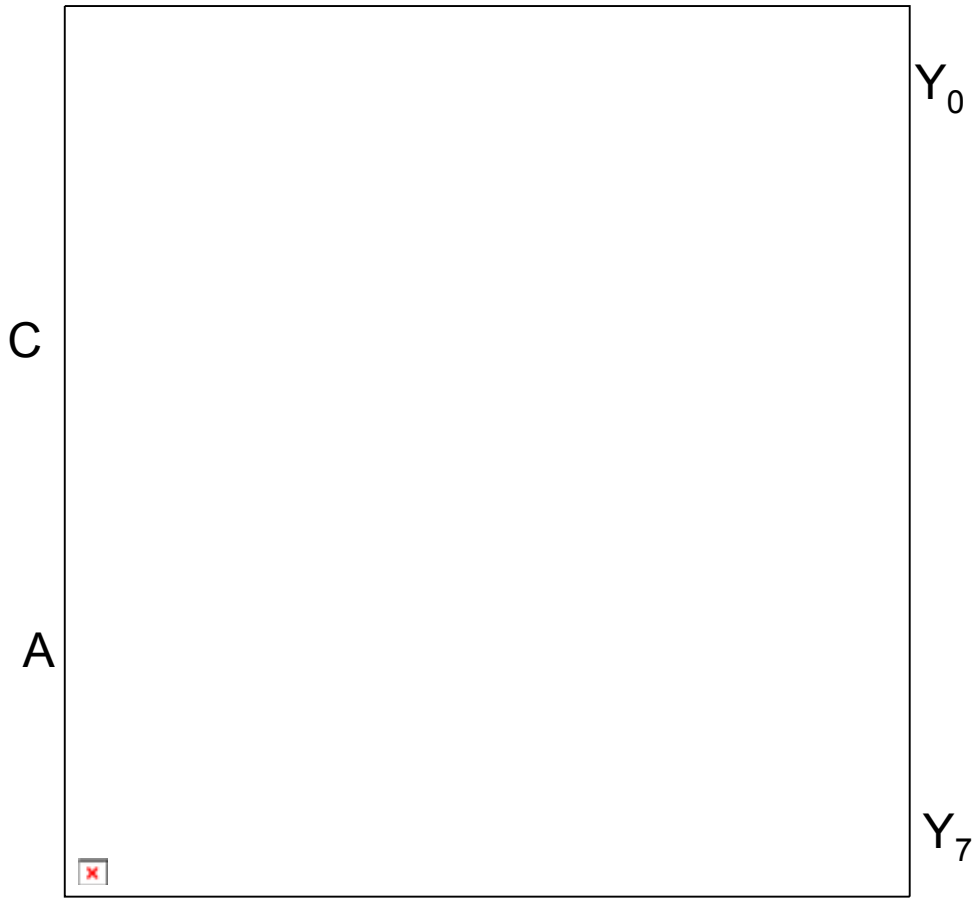


Adder / Subtractor



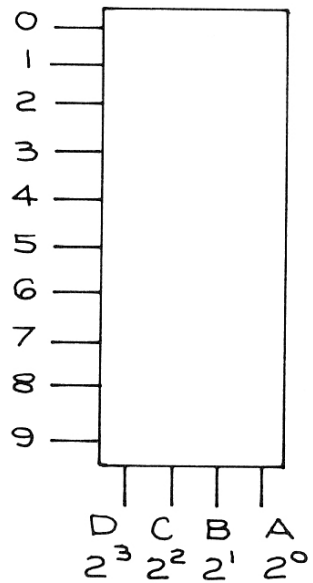
Decoder



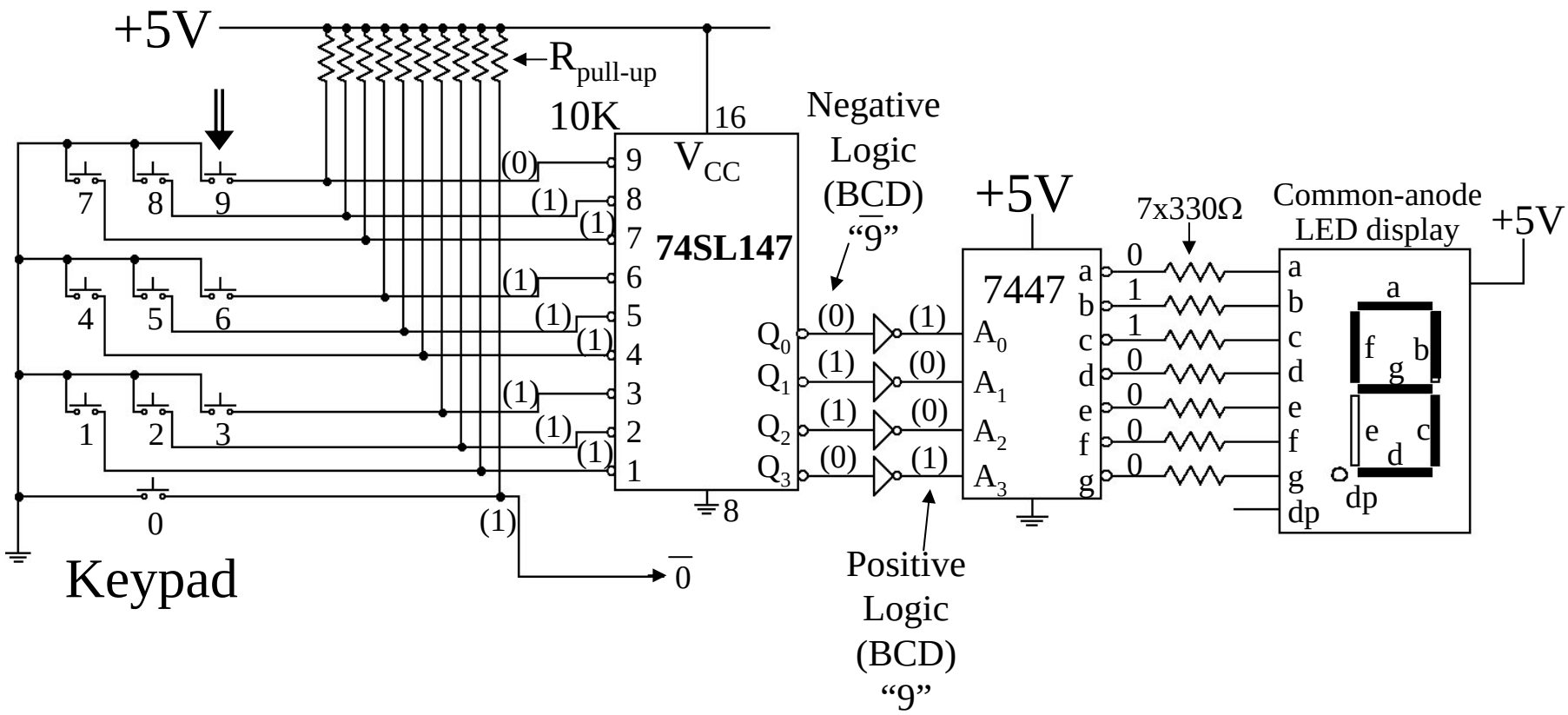


Encoder

Only one of the input lines is allowed to be active.
N input lines to log2(N) output lines.

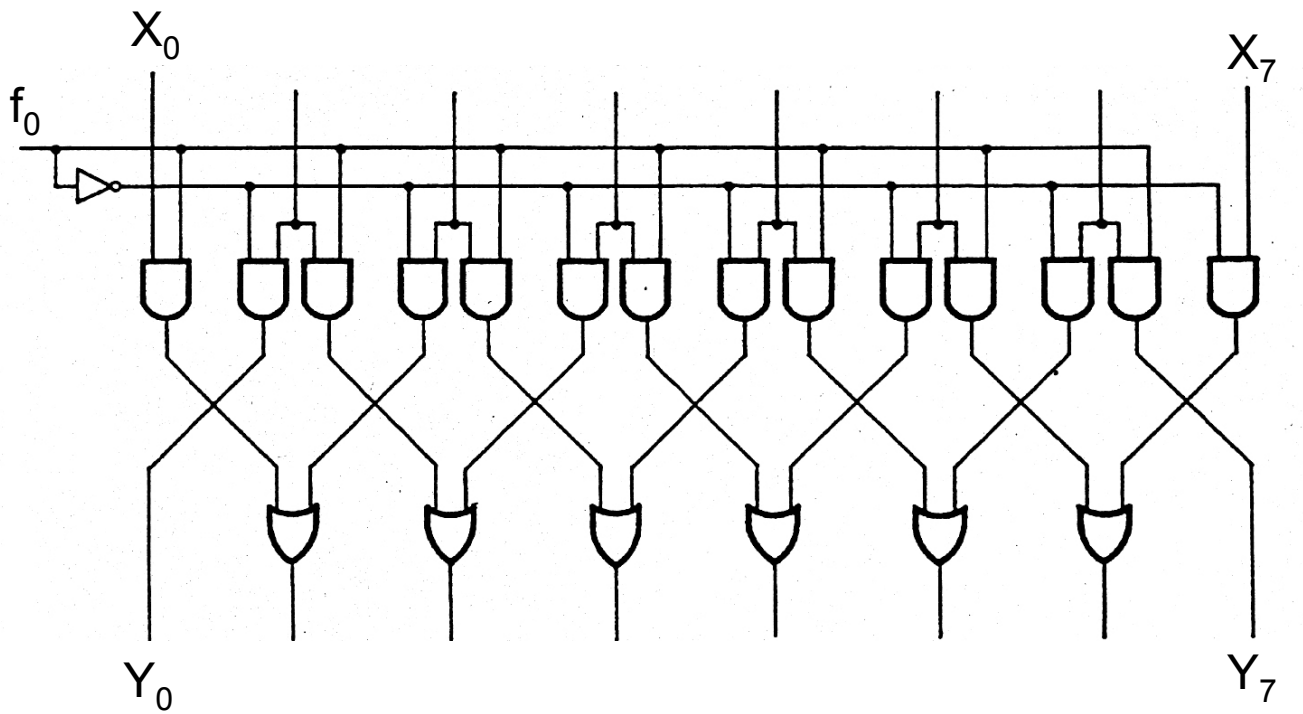


Encoder Example



Arithmetic Logical Units

- Shift Register (Division or Multiplication by 2)



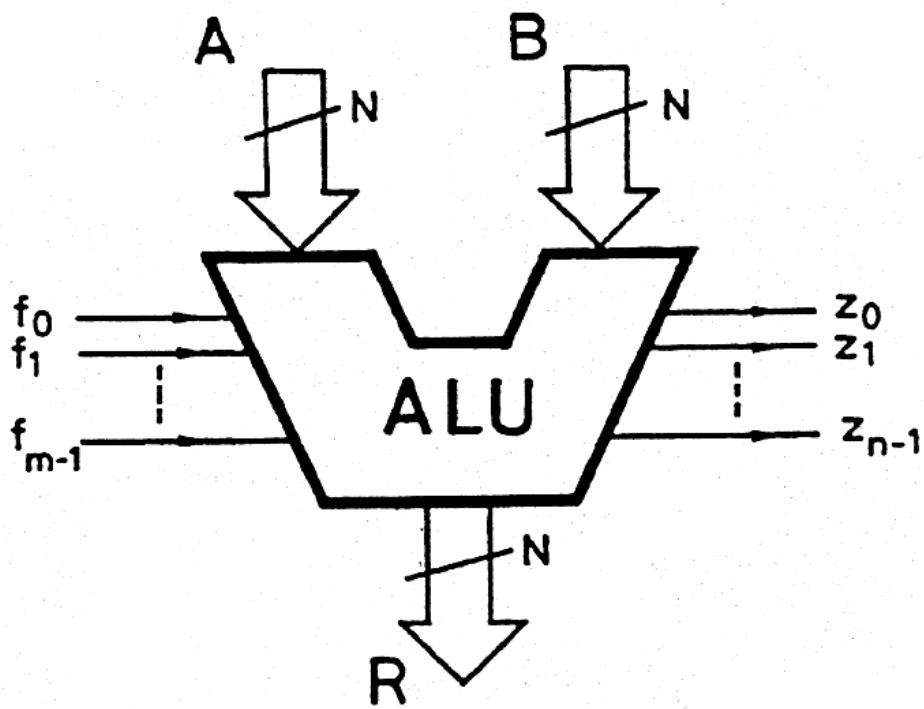
Example: X: 0110 0111

f=1 Division Y = 0011 0011

f=0 Multiplication Y = 1100 1110

Arithmetic Logical Units

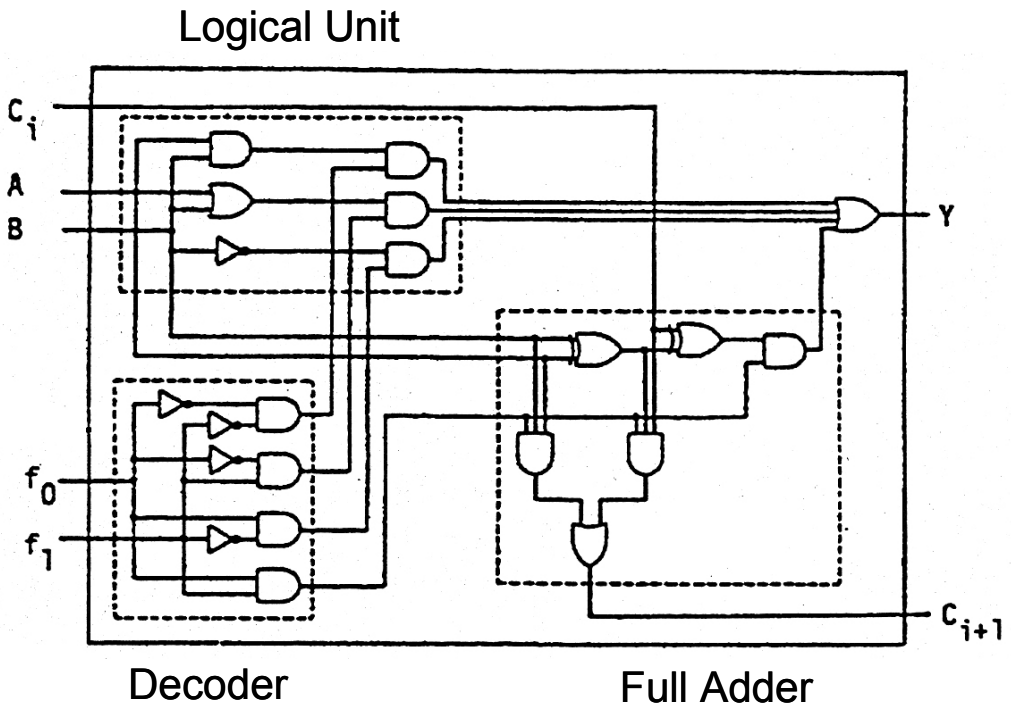
- General ALU



A, B: input
f_i: select arithmetical function
Z_i: state indicator of ALU, e.g. negative, carry, normally transmitted into condition code register

Arithmetic Logical Units

- Example simple ALU



Selected fucntion:

f_0 f_1 function

0 0 A*B

0 1 A+B

1 0 not B

1 1 A+B+Ci with carry