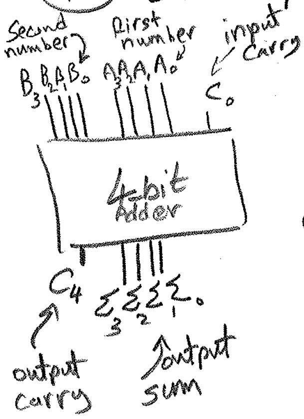
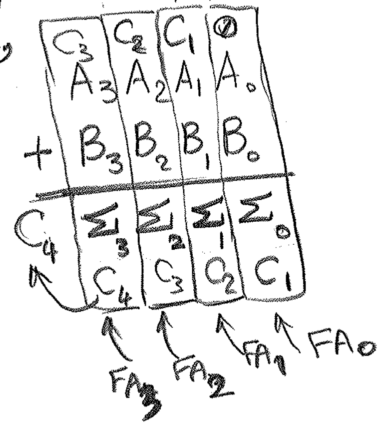
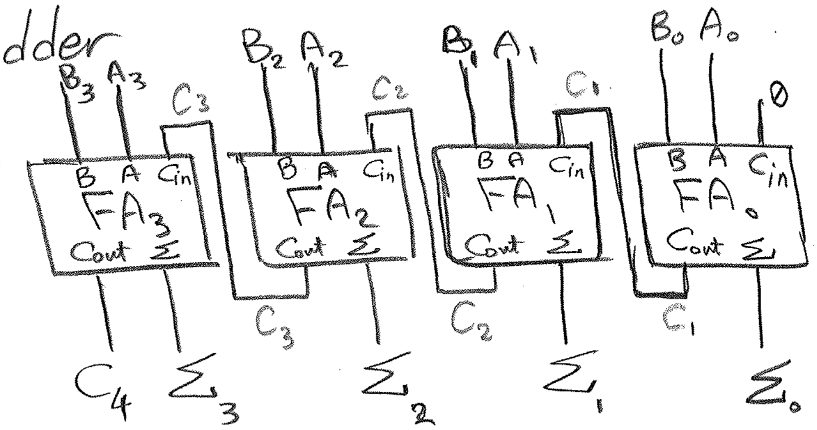
Parallel Binary Adders:

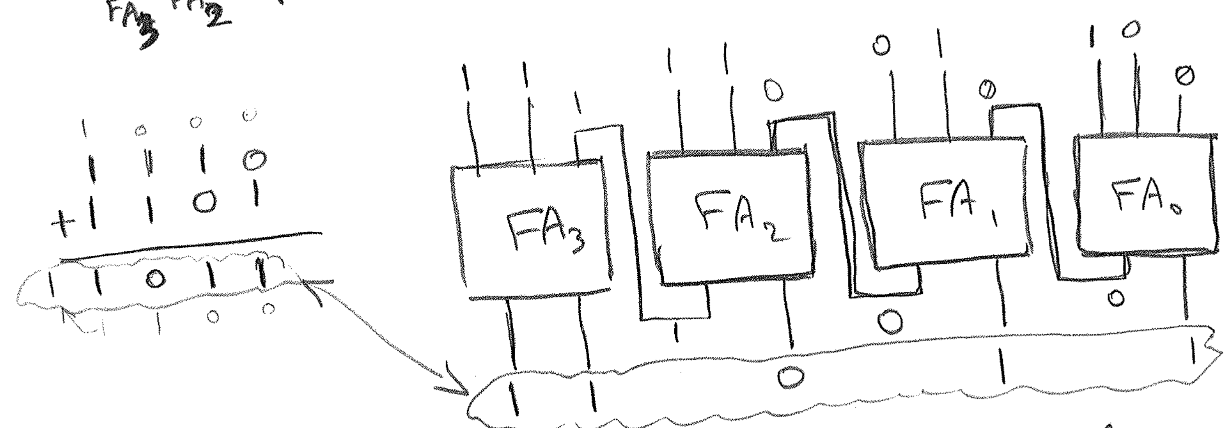
* To add two binary numbers, a **full-adder (FA)** is required for each bit.

* Example: 4-bit adder



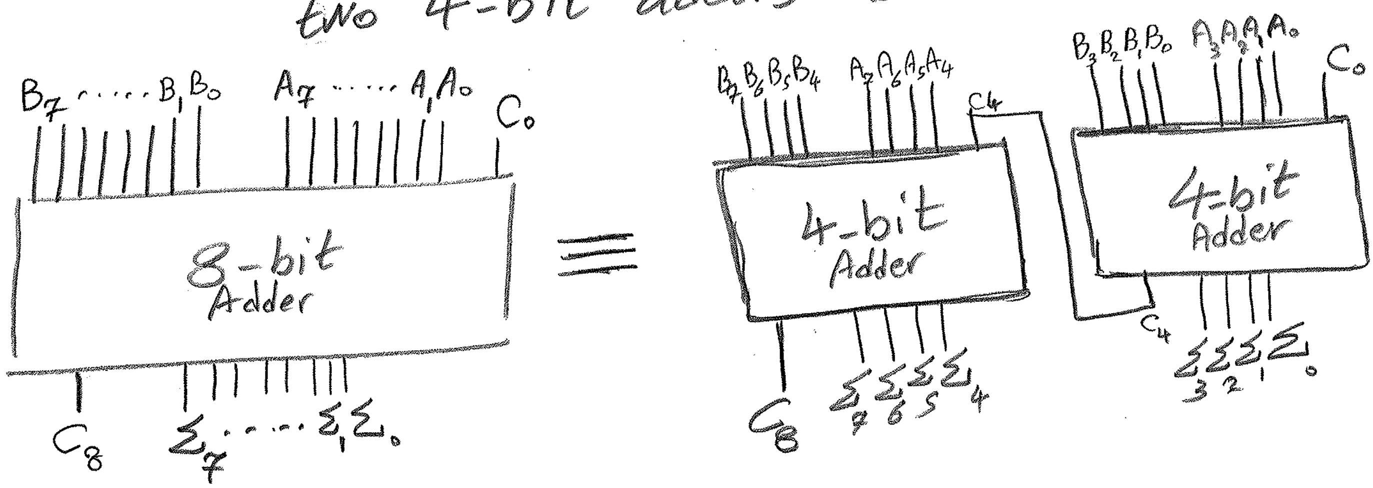




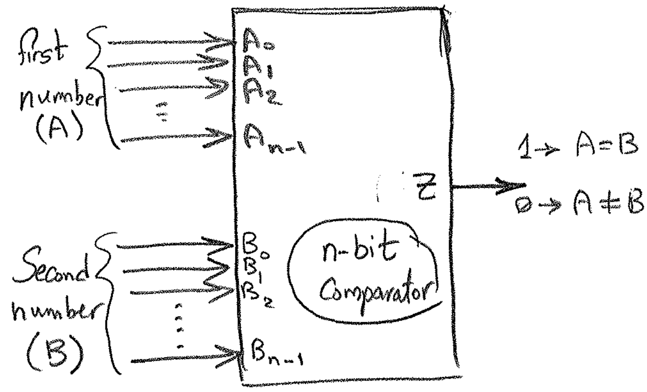


* Adders can be **cascaded** together to build bigger Adders.

* Example: 8-bit adder can be build from two 4-bit adders as follows:



Comparators :

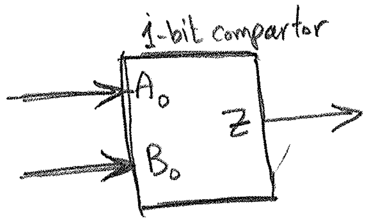


* Purpose: Compare two binary numbers A, B for equality :

$A = B \implies Z = 1$

$A \neq B \implies Z = 0$

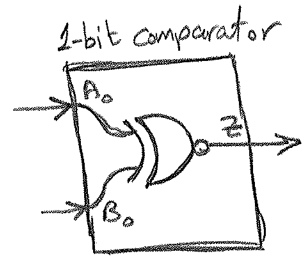
* Example: 1-bit compartor



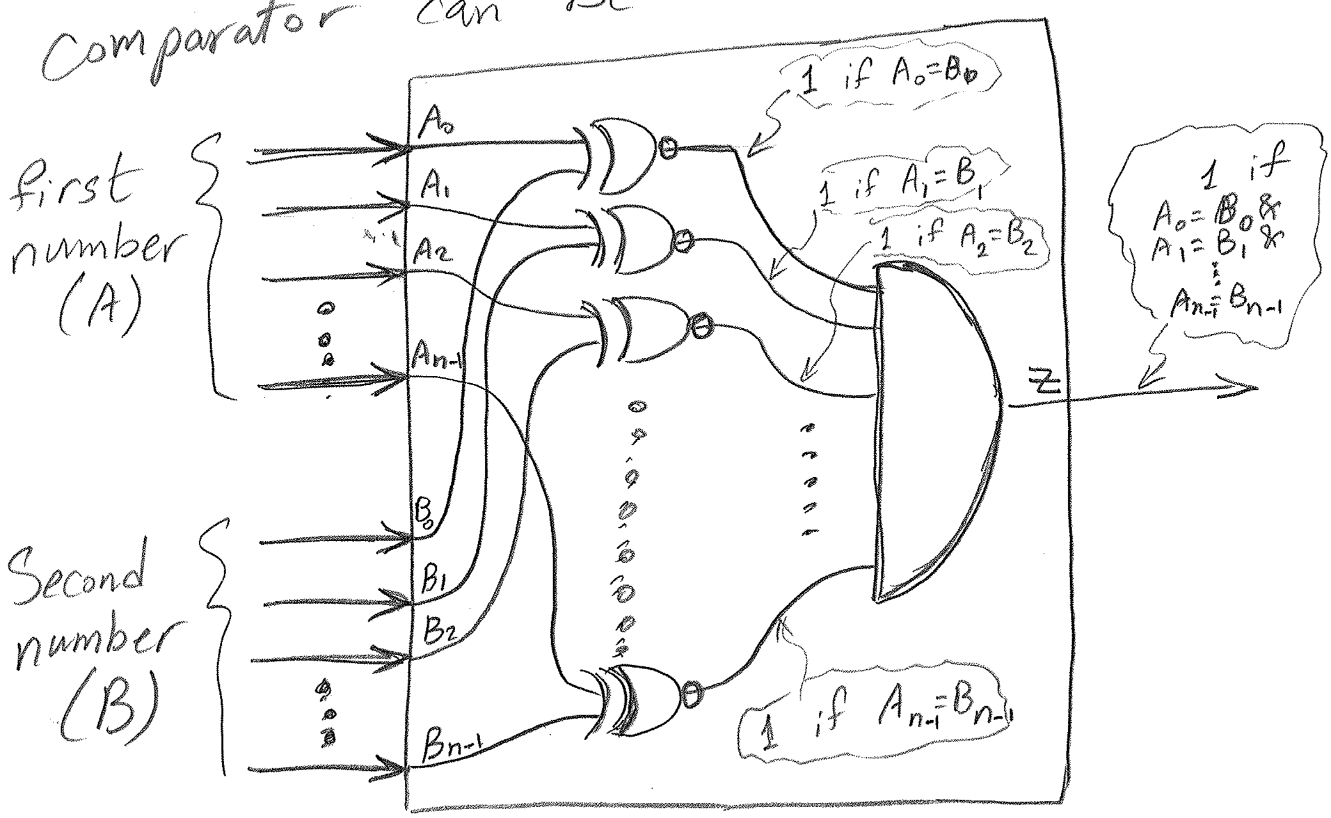
A_0	B_0	Z
0	0	1
0	1	0
1	0	0
1	1	1

XNOR

$Z = A_0 \odot B_0$

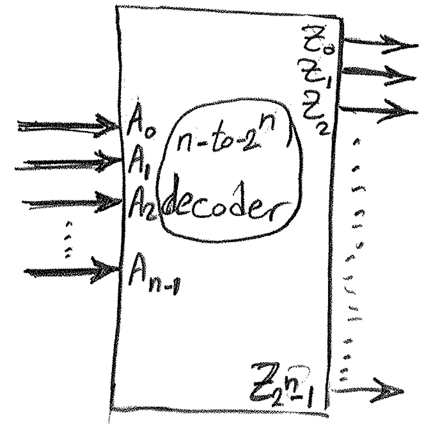


* Since an XNOR gate can be used to compare two single-bit numbers, we need **n XNOR gates** to compare the corresponding bits of two n-bit numbers. We also need an **AND gate** with **n inputs** to combine the results of the n comparisons. So, an n-bit comparator can be constructed as follows:

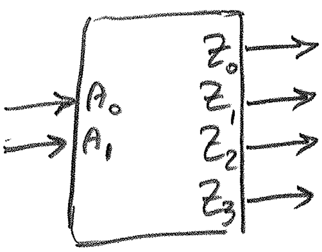


Decoders:

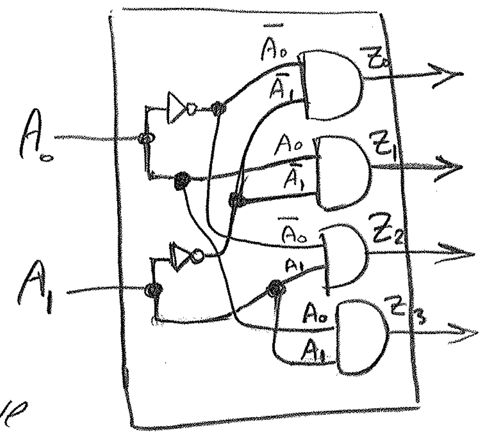
* Types: Binary decoders, BCD-to-7-segment decoder, ...etc.
* Binary decoder:
  * Commonly called $\boxed{n\text{-to-}2^n \text{ decoder}}$ (# of inputs: n; # of outputs: 2^n)
  * purpose: decodes an $\boxed{\text{n-bit binary number}}$ by activating the output which corresponds to the binary number among the $\boxed{2^n \text{ outputs}}$



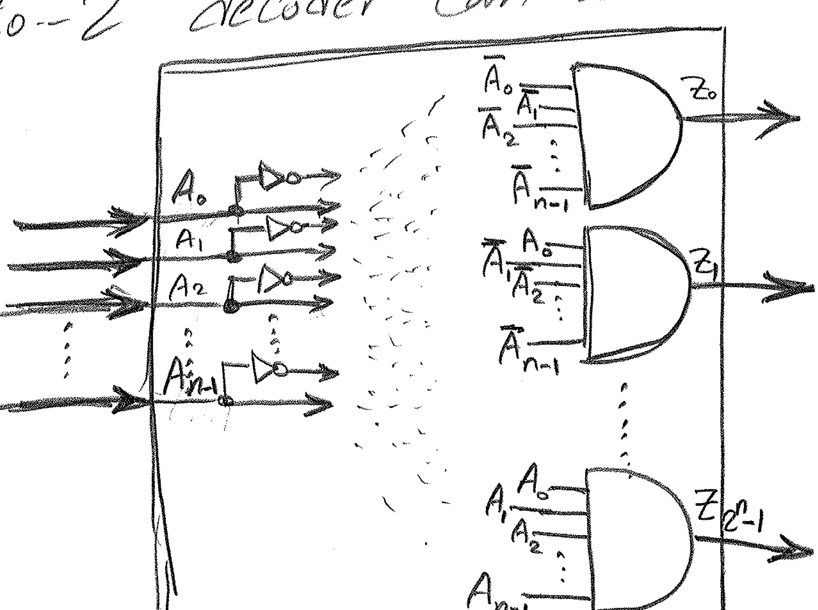
* Example: 2-to-4 decoder



A_1	A_0	Z_0	Z_1	Z_2	Z_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



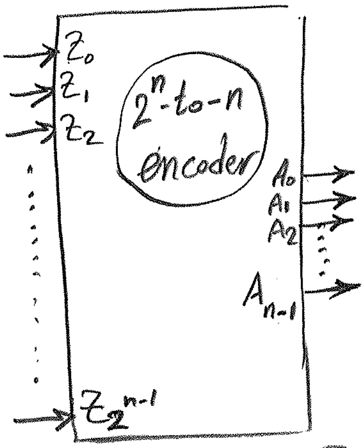
* To build an n-to-2^n decoder, we need $\boxed{2^n \text{ AND gates}}$, each AND gate must have n inputs (i.e., fan-in = n). So an n-to-2^n decoder can be constructed as follows:



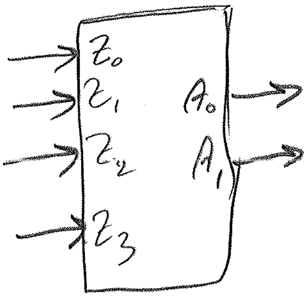
Encoders:

Types: Binary Encoders, Priority Encoders, ...etc

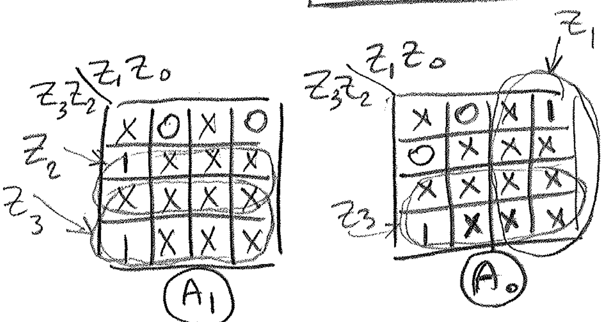
* Binary Encoders:
  * Commonly called $\boxed{2^n\text{-to-}n \text{ encoders}}$ (# of inputs: 2^n, # of outputs: n)
  * purpose: Receives 2^n bits (among which only 1 bit is active, i.e., equals 1), and sets the output to equal the binary value of the position of the active bit.



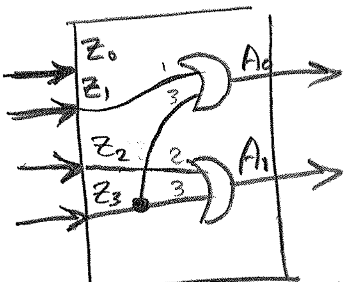
* Example: 4-to-2 encoder



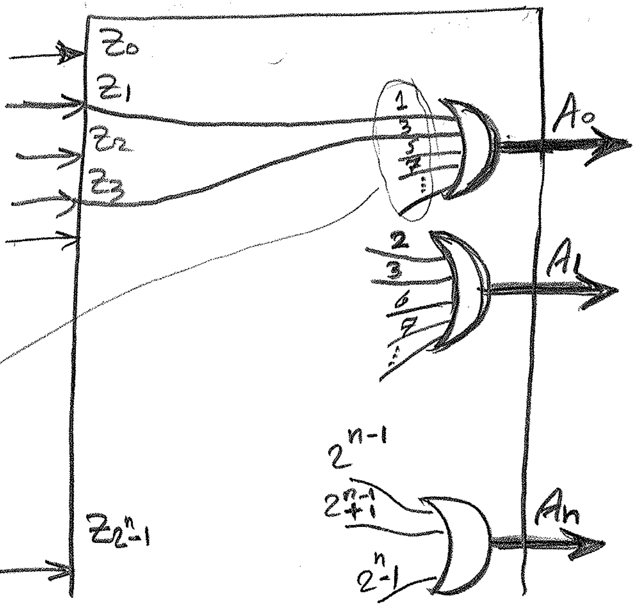
z_3	z_2	z_1	z_0	A_1	A_0
0	0	0	0	X	X
0	0	0	$\boxed{1}$	0	0
0	0	$\boxed{1}$	0	0	1
0	0	1	1	X	X
0	$\boxed{1}$	0	0	1	0
	⋮			X	X
$\boxed{1}$	0	0	0	1	1
	⋮			X	X
1	1	1	1		



$A_1 = z_2 + z_3$ $A_0 = z_1 + z_3$



* To build a 2^n-to-n encoder, we need $\boxed{n \text{ OR gates}}$



Note: $A_0 = 1$ when position equals:

1 ⟶ 00...00$\boxed{1}$

or 3 ⟶ 00...01$\boxed{1}$

or 5 ⟶ 00...10$\boxed{1}$

or 7 ⟶ 00...11$\boxed{1}$