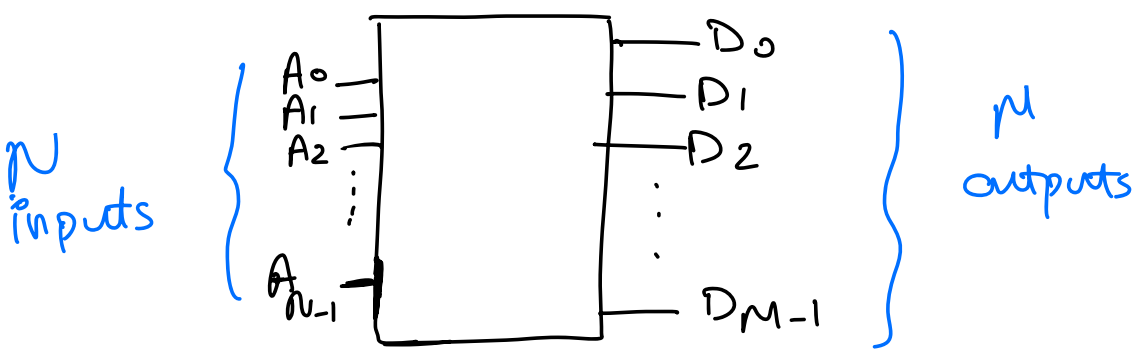
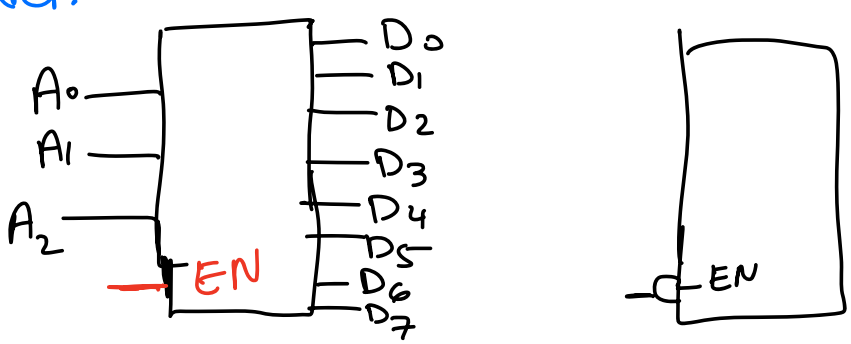
Transistors → gates (AND, OR, NAND, ⋮) → Specialized building blocks (Decoders, encoders, multiplexers, ..)

N-to-M Binary Decoder:



3-to-8 Decoder:

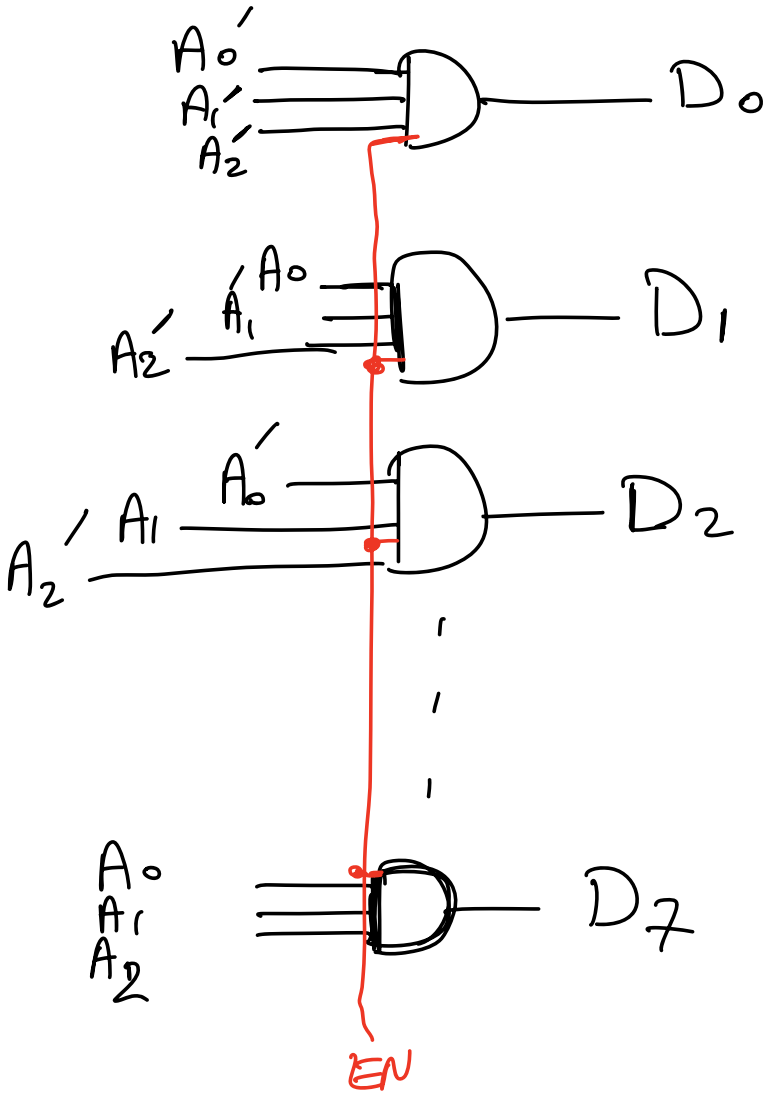


The value of the input makes the corresponding output to be true:

$$\begin{cases} A_0 = 0 \\ A_1 = 0 \\ A_2 = 0 \end{cases} \longrightarrow \begin{cases} D_0 = 1 \\ D_1, \ldots, D_7 = 0 \end{cases}$$

$$\begin{cases} A_0 = 1 \\ A_1 = 0 \\ A_2 = 0 \end{cases} \longrightarrow \begin{cases} D_0 = 0, \; D_1 = 1, \; D_2, \ldots, D_7 = 0 \end{cases}$$

A_2	A_1	A_0	EN	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
X	X	X	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0
1	0	1	1	0	0	1	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

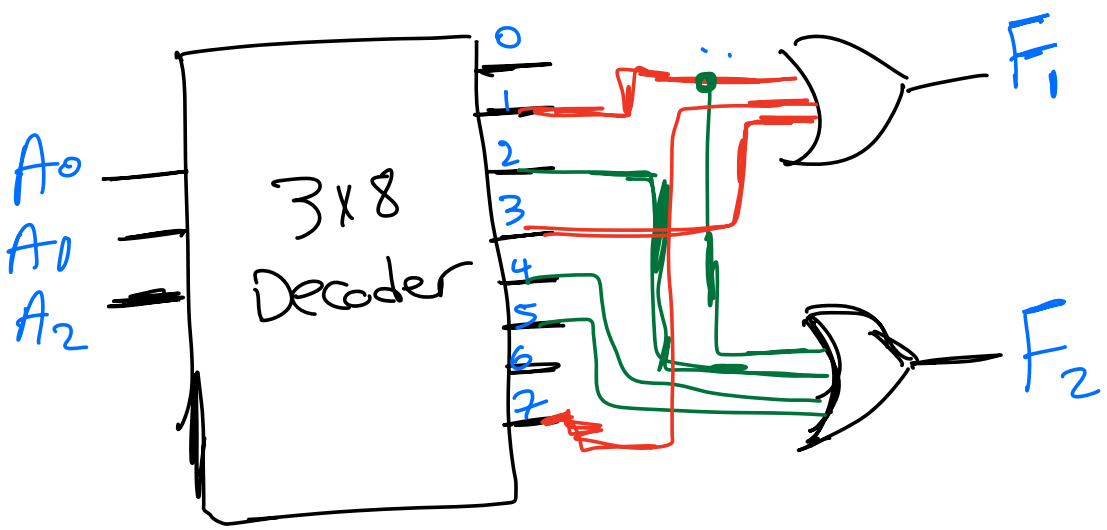


Any Boolean function can be implemented using a decoder and "OR" gates.

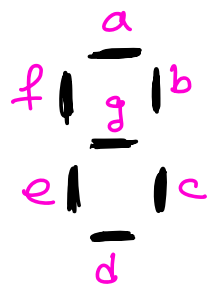
Example: implement F_1 and F_2 using a 3x8 decoder and two OR gates.

A_2	A_1	A_0	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

Similar to Sum of minterms

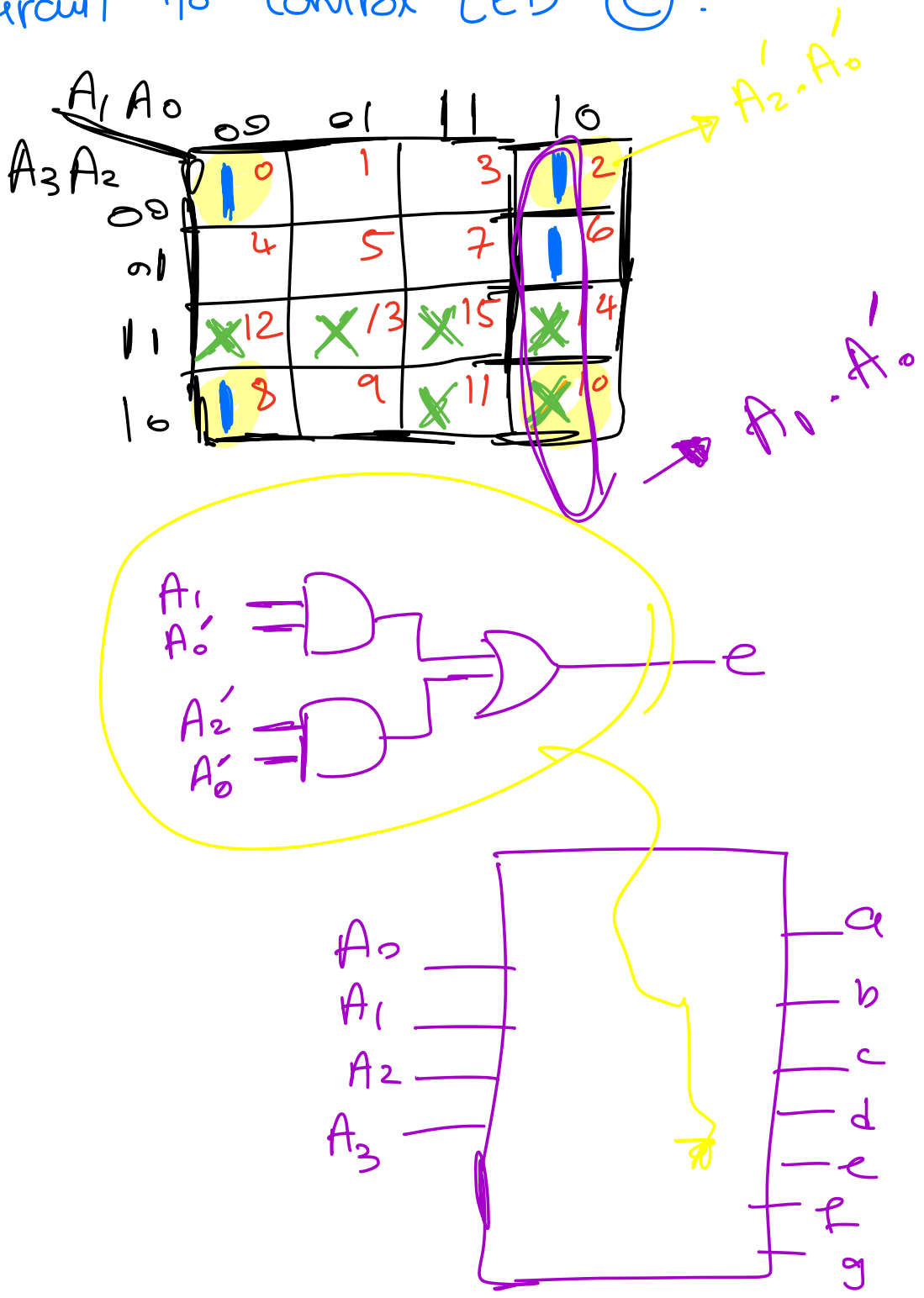


BCD-to 7-Segment decoder



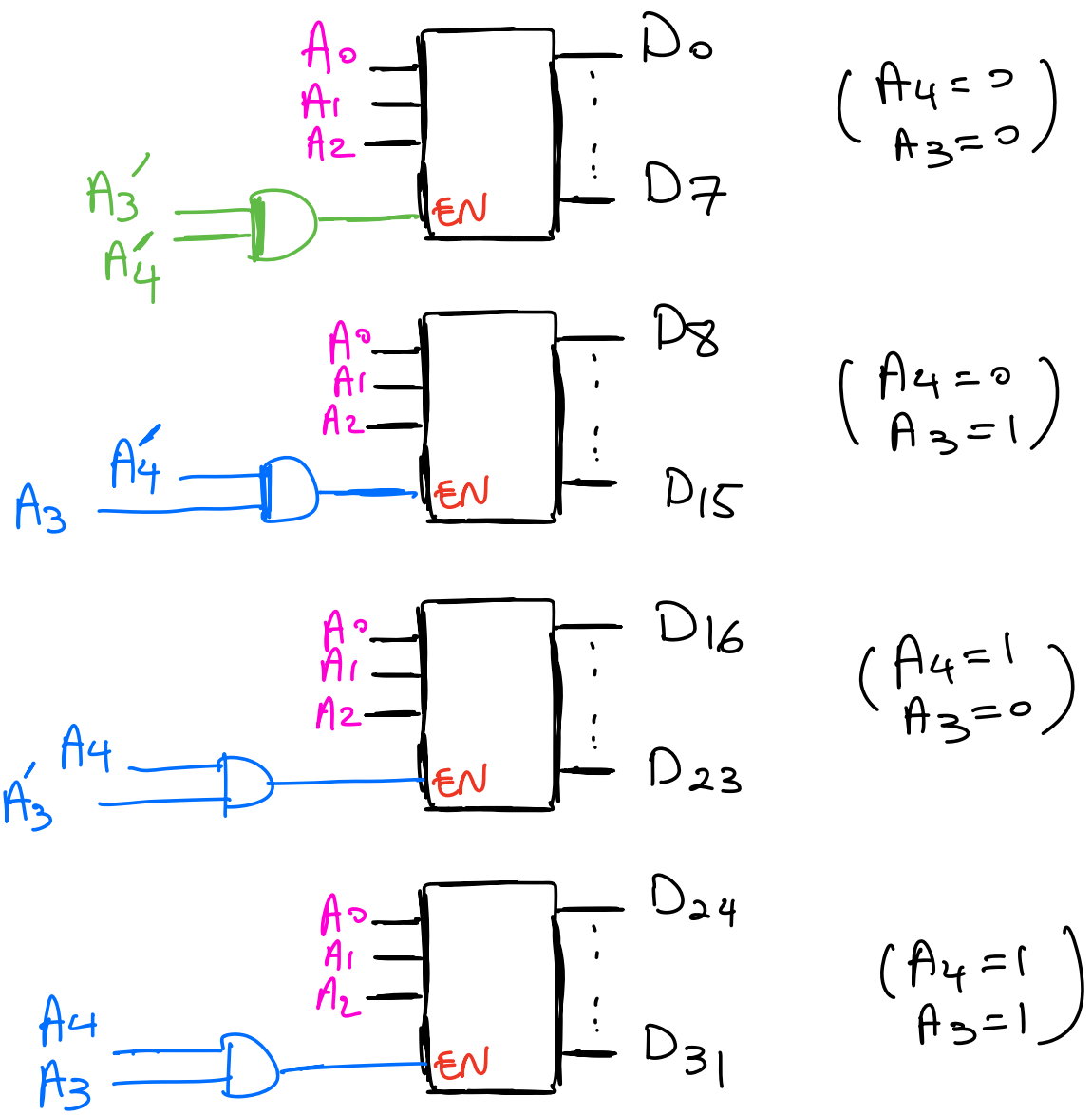
A_3	A_2	A_1	A_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1

Circuit to Control LED (e):



Cascading Decoders:

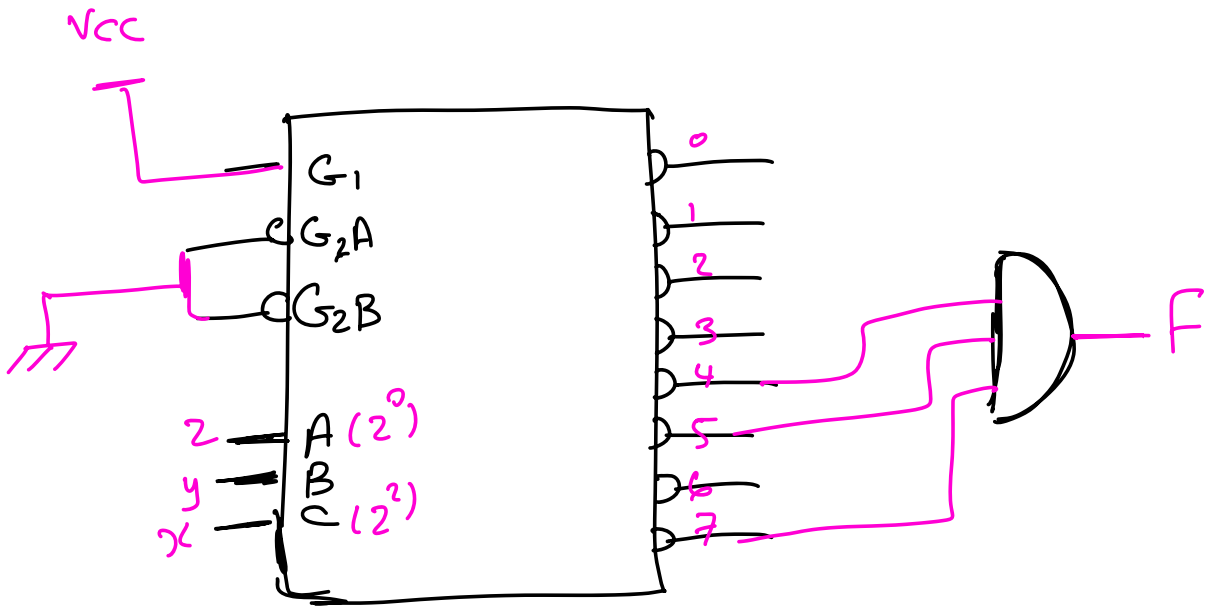
Example: Design a 5-to-32 Decoder using 3-to-8 Decoders, which have "EN" input.



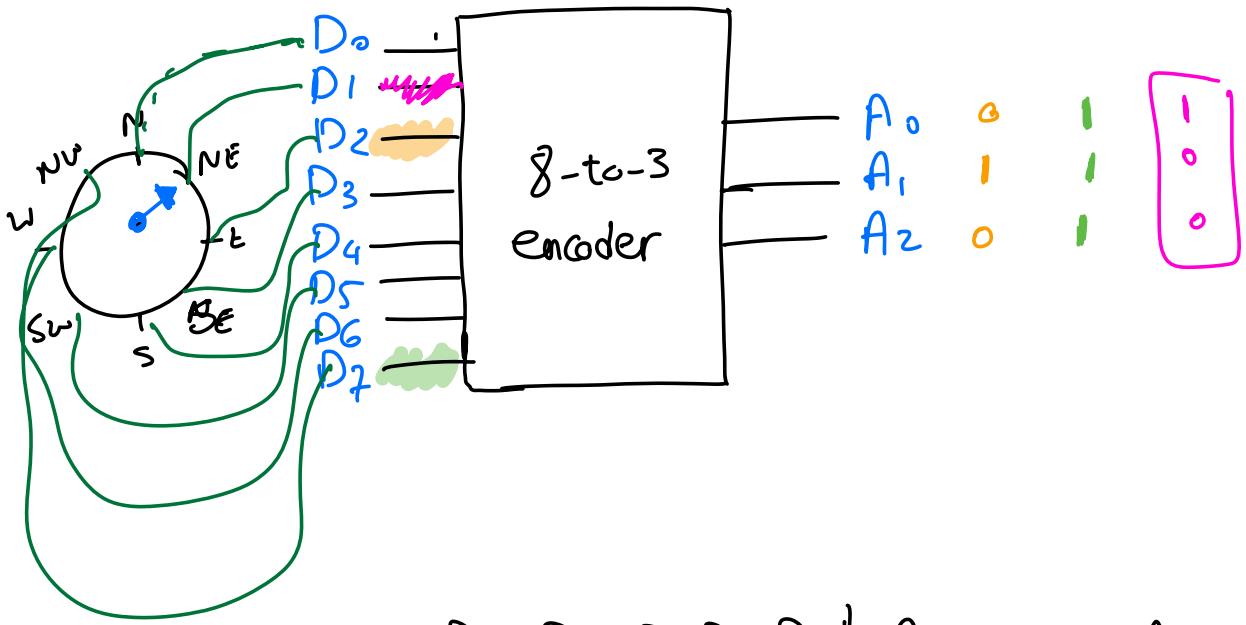
$A_4\ A_3\ A_2\ A_1\ A_0$

Use the Decoder 74X138 to realize:

$$f(x, y, z) = \prod(4, 5, 7)$$



Encoders:



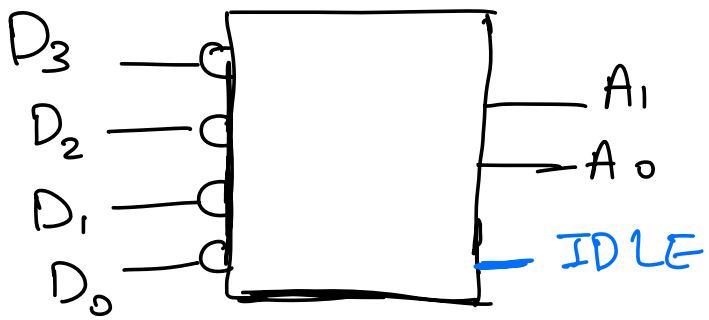
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
.										
.										
.										
1	0	0	0	0	0	0	0	1	1	1

If more than one input is "1", then the encoder does not work properly.

Priority Encoders

Example: Design a 4-to-2 priority encoder that gives "priority" to the most significant input position:

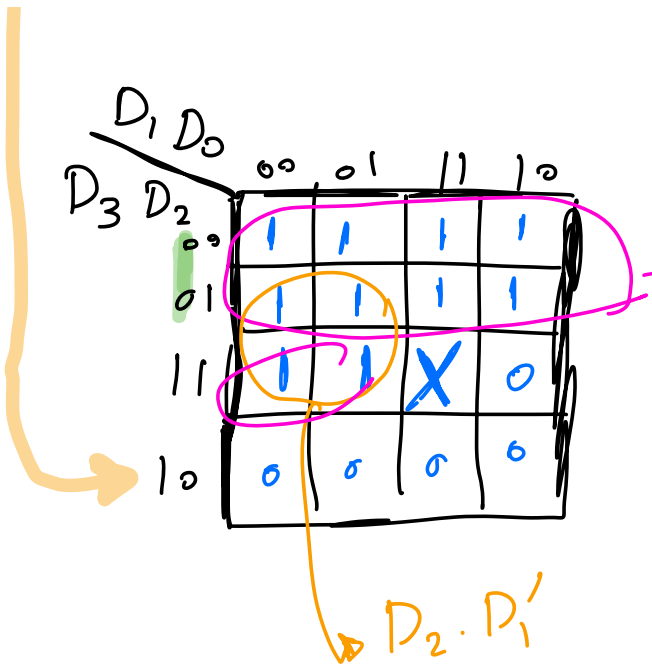
$$D_3 > D_2 > D_1 > D_0$$



IDLE = "1" if the encoder is not used.

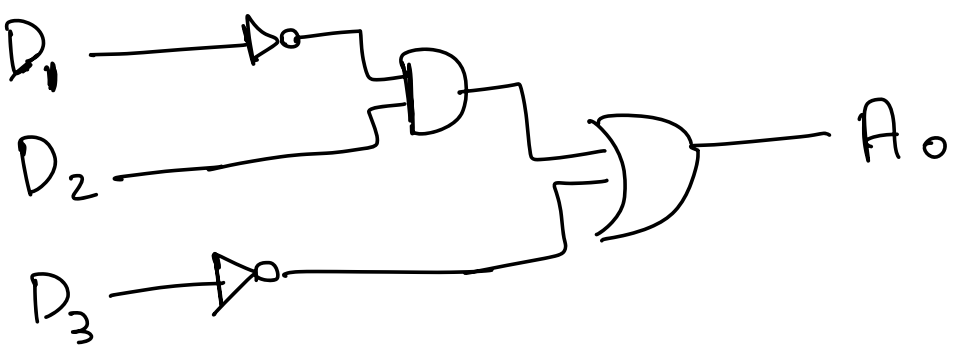
IDLE = "0" if the encoder is used.

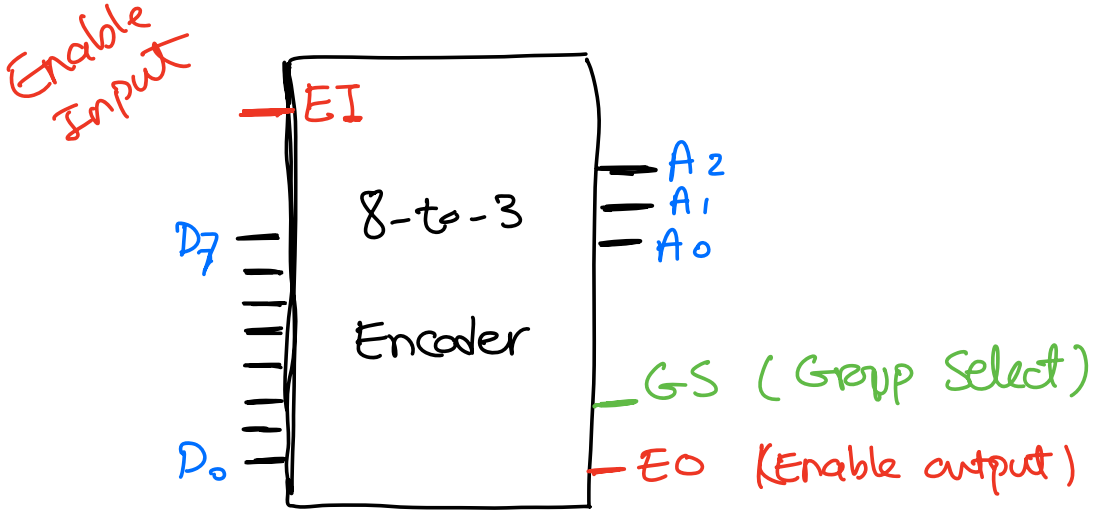
D_3	D_2	D_1	D_0	A_1	A_0	IDLE
1	1	1	0	0	0	0
1	1	0	X	0	1	0
1	0	X	X	1	0	0
0	X	X	X	1	1	0
1	1	1	1	X	X	1



This is for output "A0".

$$A_0 = D_3' + D_2 . D_1'$$



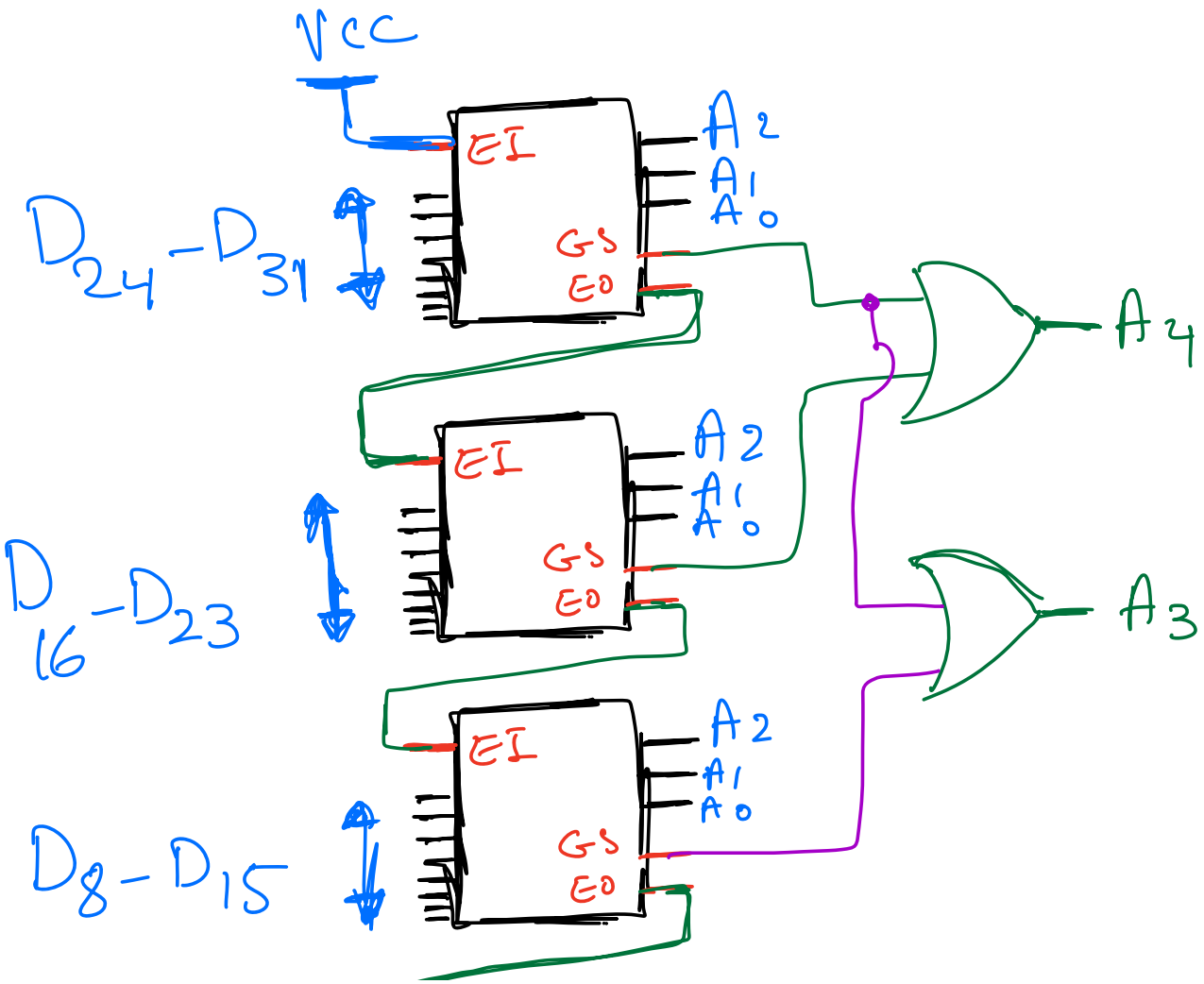


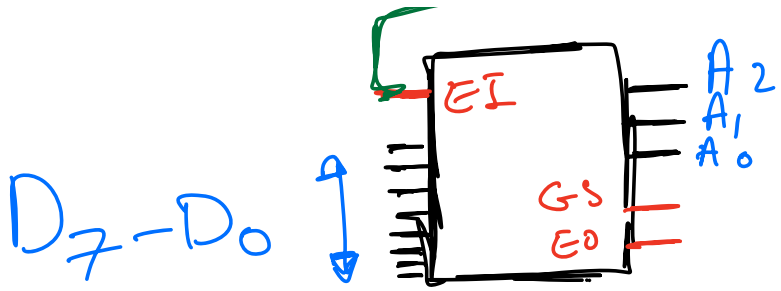
GS: { EI = 1 AND one or more inputs are asserted. } => GS=1

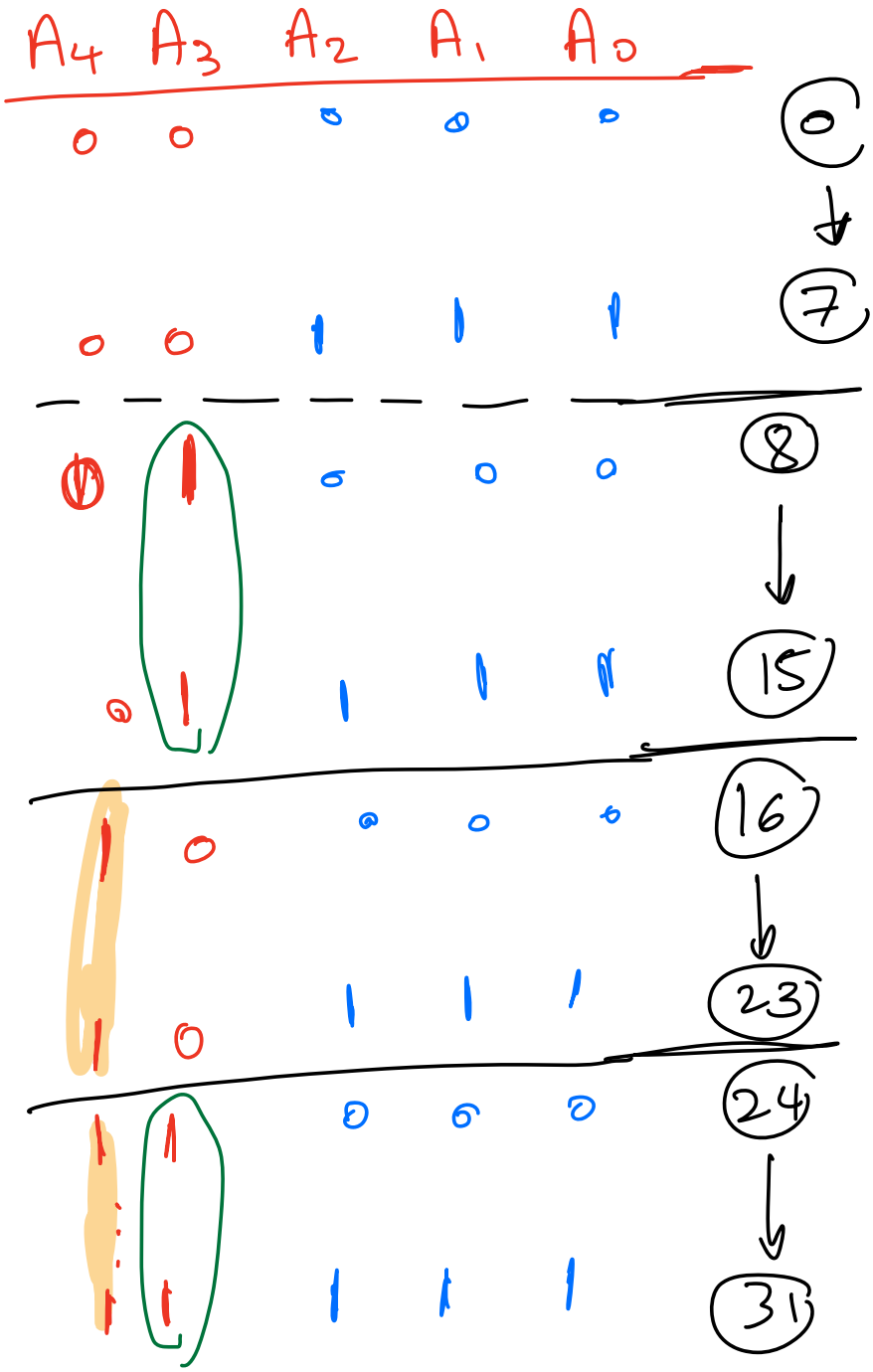
EO = 1 if EI=1 but there is nothing at the input

---

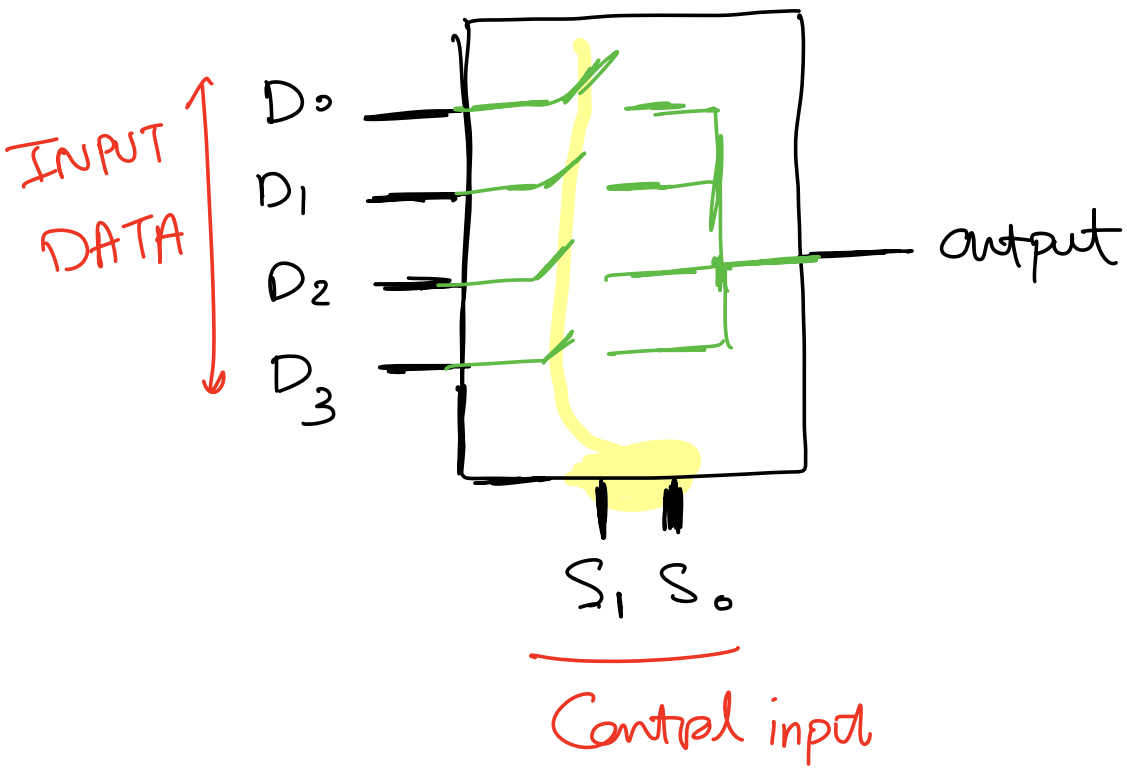
build a 32-to-5 encoder using 8-to-3 encoders with EI, EO, and GS input/outputs.




D7-D0
EI
A2
A1
A0
GS
EO


A4 A3 A2 A1 A0
0
7
8
15
16
23
24
31

Multiplexers : (Digital Switch)

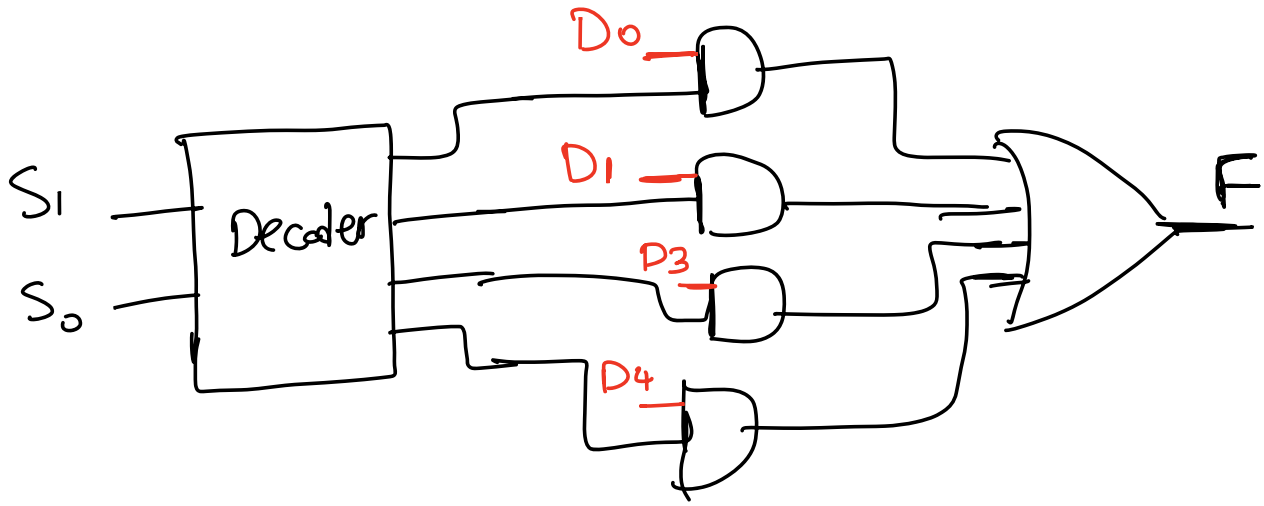


$S_1 = 0 \quad S_0 = 0 \Rightarrow$ output $= D_0$

$S_1 = 0, S_0 = 1 \Rightarrow$ output $= D_1$

S_1 S_0	output
0 0	D_0
0 1	D_1
1 0	D_2
1 1	D_3

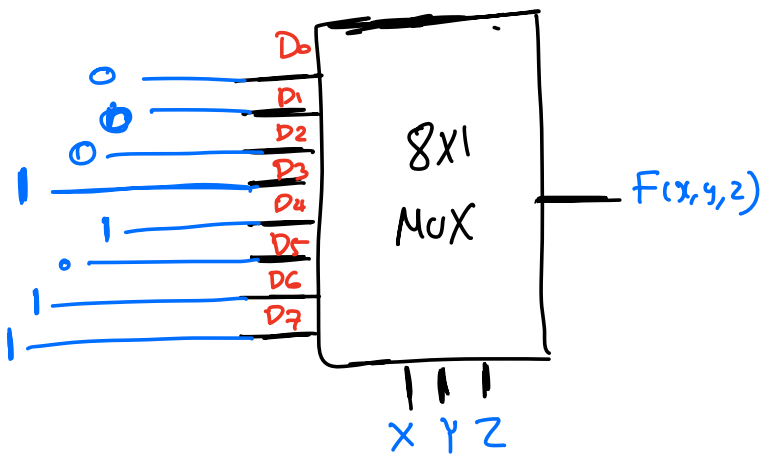
S_1 \ S_0	0	1
0	D_0	D_1
1	D_2	D_3



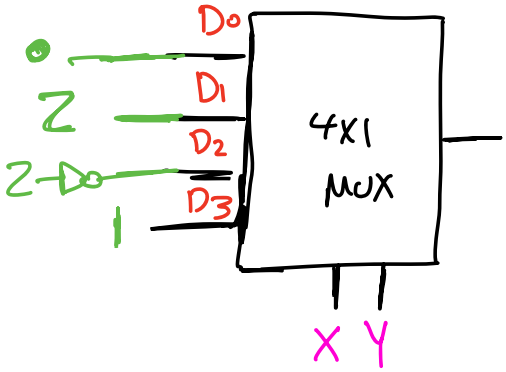
Building functions with multiplexers:

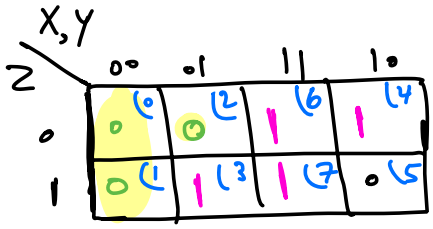
$$f(x,y,z) = \sum(3,4,6,7)$$

Build $f(x,y,z)$ with an 8 x 1 multiplexer:



Build F(x,y,z) using a 4x1 multiplexer:

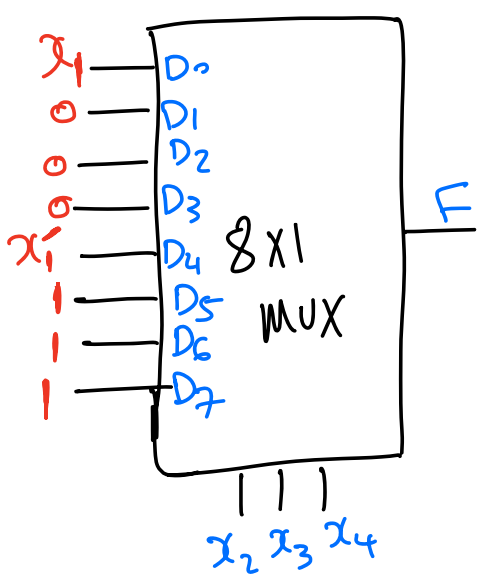


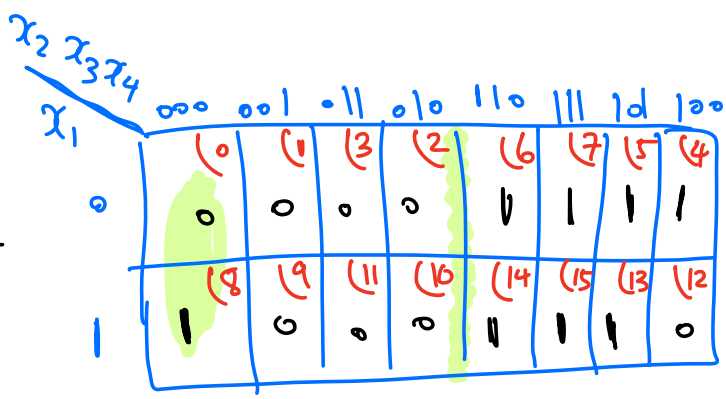


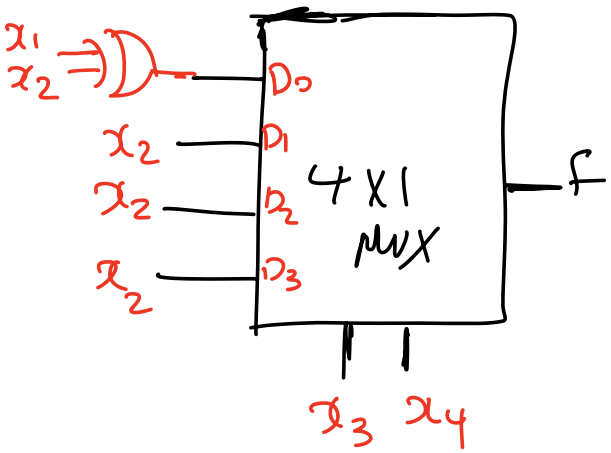
Example: $f(x_1, x_2, x_3, x_4) = \sum(4, 5, 6, 7, 8, 13, 14, 15)$

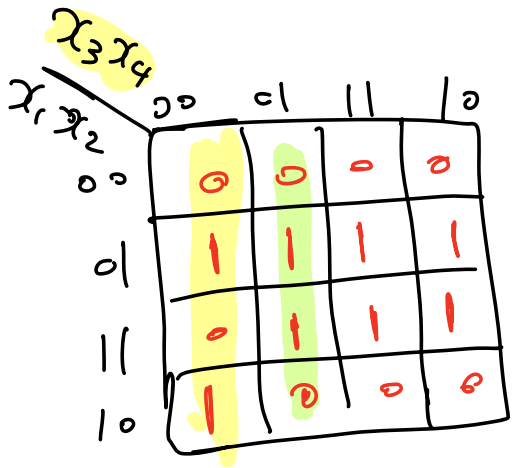
(a) build with an 8x1 MUX

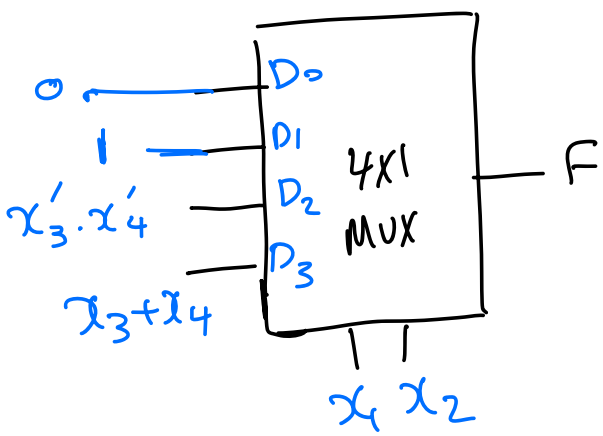
(b) build with a 4x1 MUX

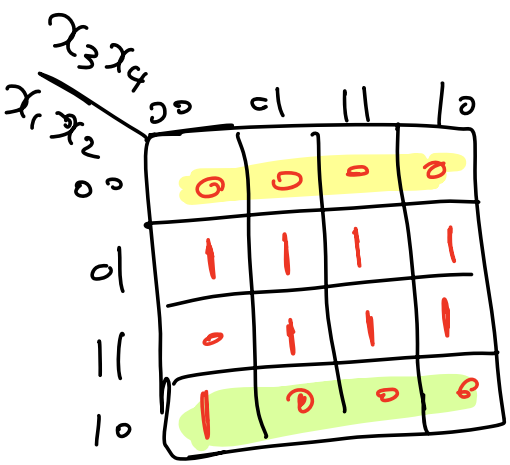








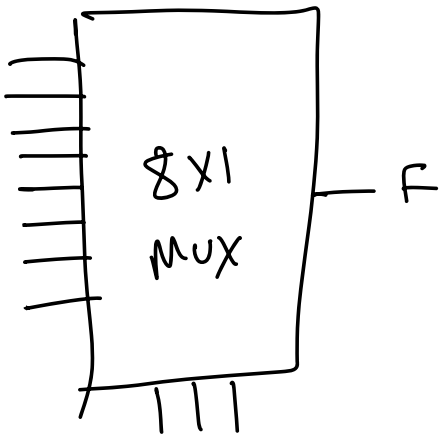


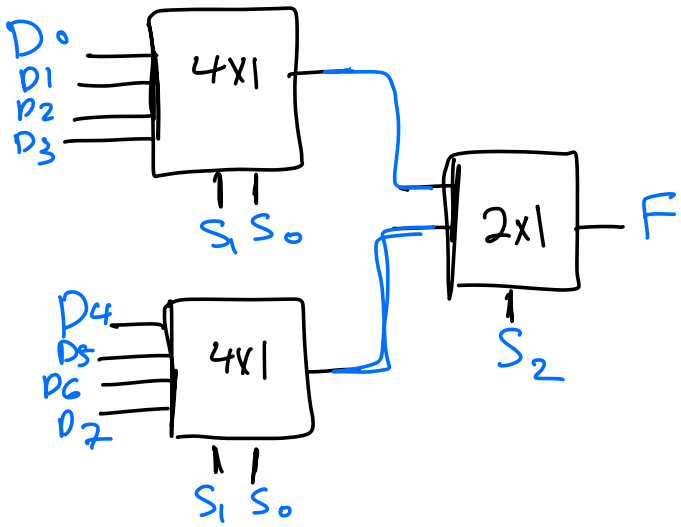


$(x_3+x_4)' = x_3 \odot x_4 = D_3$

Build an 8x1 MUX using two 4x1 MUX an

a 2x1 MUX

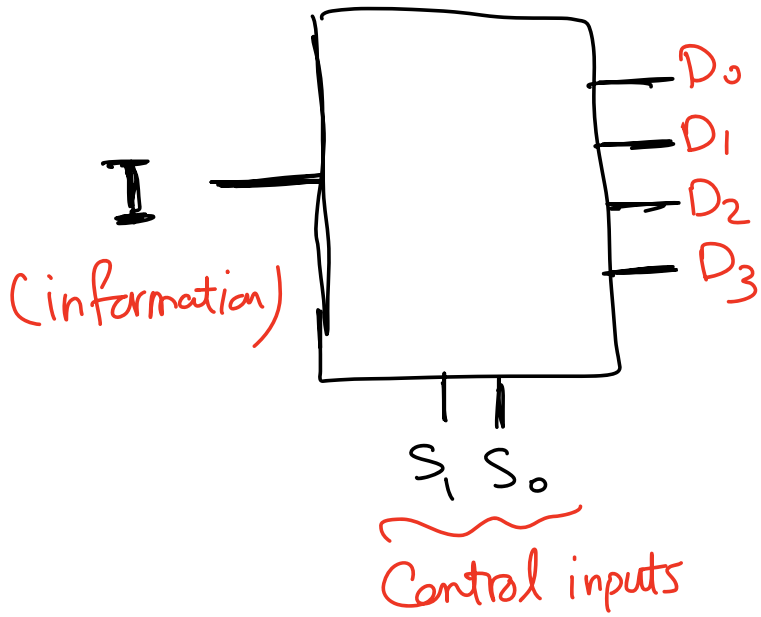




S_2	S_1	S_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Build an 8x1 MUX using four 2x1 MUX and one 4x1 MUX.

DeMultiplexers :



$S_1 = 0\ S_0 = 0 \Rightarrow D_0 = I$

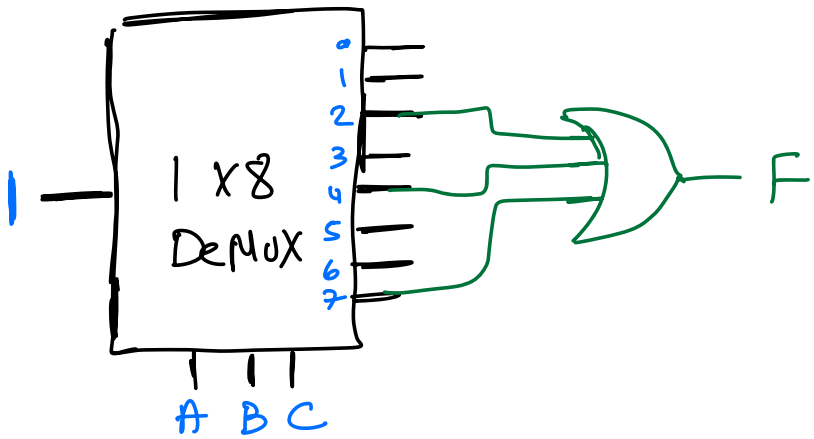
$S_1 = 0, S_0 = 1 \Rightarrow D_1 = I$

$S_1 = 1, S_0 = 0 \Rightarrow D_2 = I$

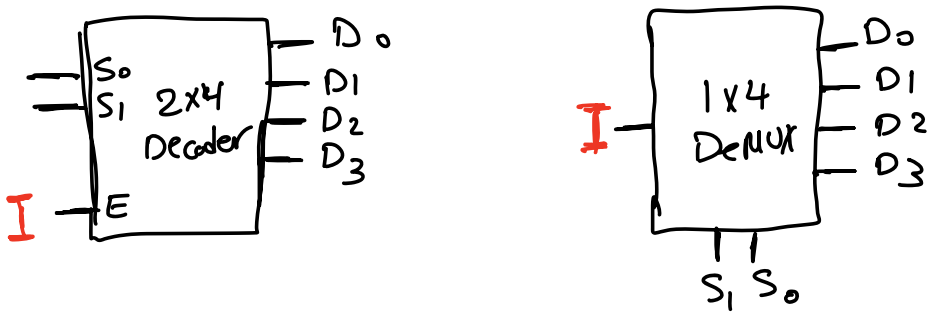
$S_1 = 1, S_0 = 1 \Rightarrow D_3 = I$

Build functions with Demultiplexers

$$f(A, B, C) = \sum(2, 4, 7)$$



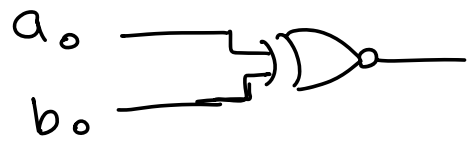
Convert a decoder to demultiplexer:



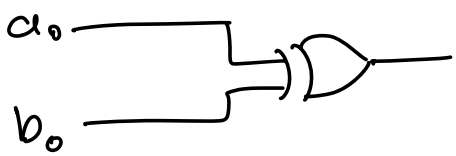
Comparators:

- **Equality Comparators**: Compares two binary #s, and indicates if they are equal.
- **Magnitude Comparators**: Compares two binary #s, and produces relational conditions: $A<B$, $A=B$, $A>B$

How to build equality Comparators:



active-high output when the two inputs are equal.

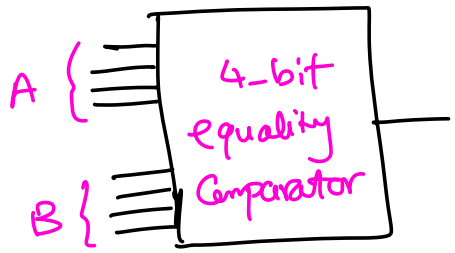


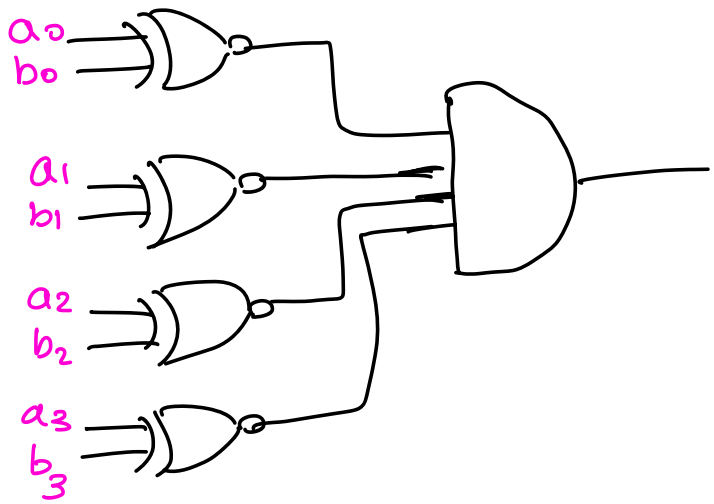
active-Low output when the the two inputs are equal.

4-bit equality Comparator:

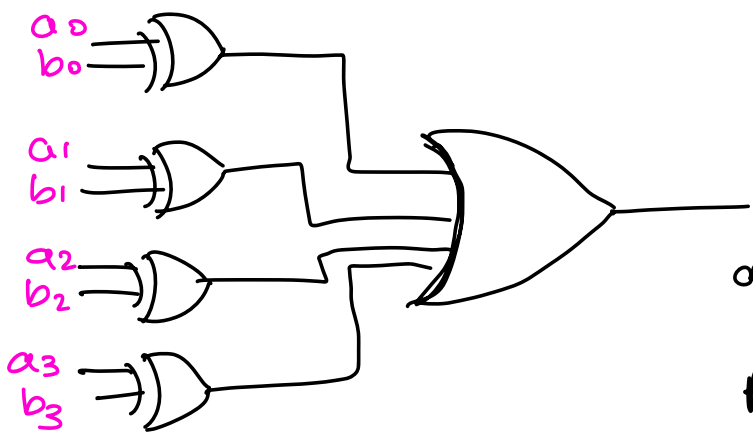
A: $a_3\ a_2\ a_1\ a_0$

B: $b_3\ b_2\ b_1\ b_0$





active-high output when A and B are equal.



active-low output when A and B are equal.

Magnitude Comparator : $\begin{cases} A<B \\ A=B \\ A>B \end{cases}$

$A=(93)_{10}$
$B=(78)_{10}$
$\Big)$ $9>7 \Rightarrow A>B$

Algorithm:

- Start with the most Significant digit and do Comparison
- Keep Comparing digits to the right
- Once any digit of "A" is greater or less than the corresponding digit in "B", then we can make a decision.
- If all digits are equal, then both A and B are equal.

$A: a_3\ a_2\ a_1\ a_0$, $B: b_3\ b_2\ b_1\ b_0$

a_3, b_3	a_2, b_2	a_1, b_1	a_0, b_0	A>B	A<B	A=B
$a_3 > b_3$	X	X	X	1	0	0
$a_3 < b_3$	X	X	X	0	1	0
$a_3 = b_3$	$a_2 > b_2$	X	X	1	0	0
$a_3 = b_3$	$a_2 < b_2$	X	X	0	1	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 > b_1$	X	1	0	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 < b_1$	X	0	1	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 = b_1$	$a_0 > b_0$	1	0	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 = b_1$	$a_0 < b_0$	0	1	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 = b_1$	$a_0 = b_0$	0	0	1

For $a_3 > b_3$ to be true: $\begin{cases} a_3 = 1 \\ b_3 = 0 \end{cases}$

$$a_3 \cdot b_3'$$

For $a_3 = b_3$ AND $a_2 > b_2$ to be true

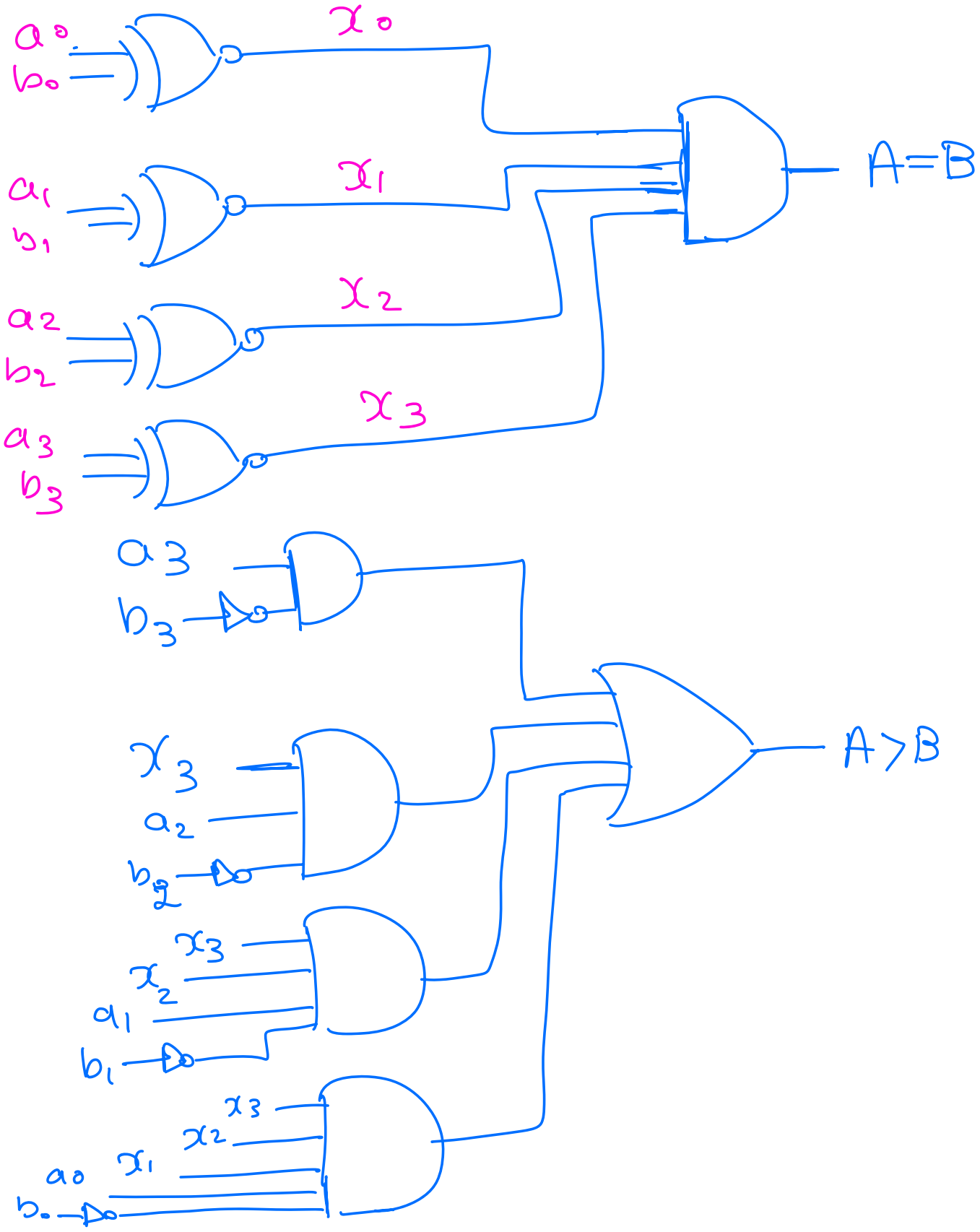
$$(a_3 \odot b_3) \cdot (a_2 \cdot b_2')$$

↑ XNOR

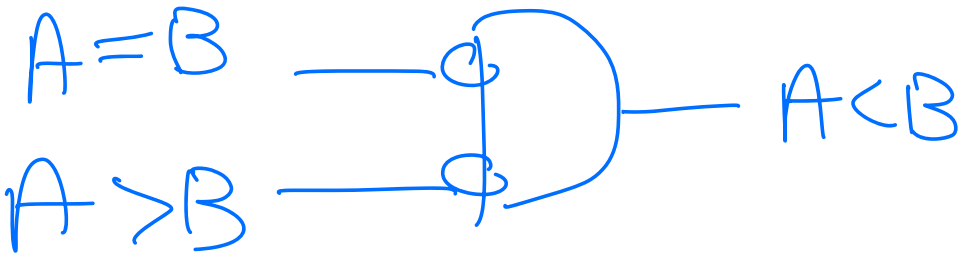
$$A > B : \; a_3 \cdot b_3' + [(a_3 \odot b_3) \cdot (a_2 \cdot b_2')]$$

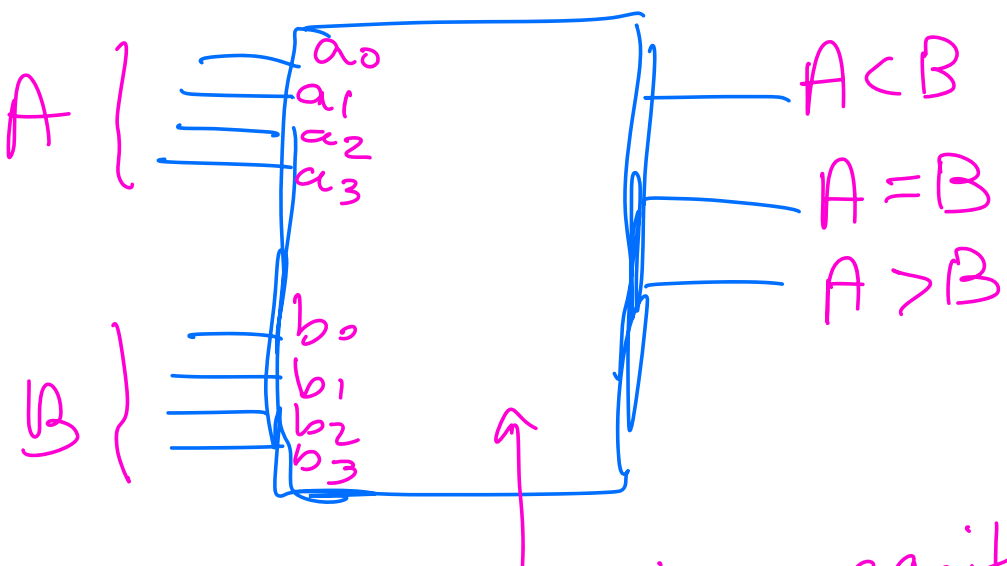
$$+ [(a_3 \odot b_3) \cdot (a_2 \odot b_2) \cdot (a_1 \cdot b_1')]$$

$$+ [(a_3 \odot b_3) \cdot (a_2 \odot b_2) \cdot (a_1 \odot b_1) \cdot (a_0 \cdot b_0')]$$


a_0
b_0
x_0
a_1
b_1
x_1
a_2
b_2
x_2
a_3
b_3
x_3
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
A>B

A<B



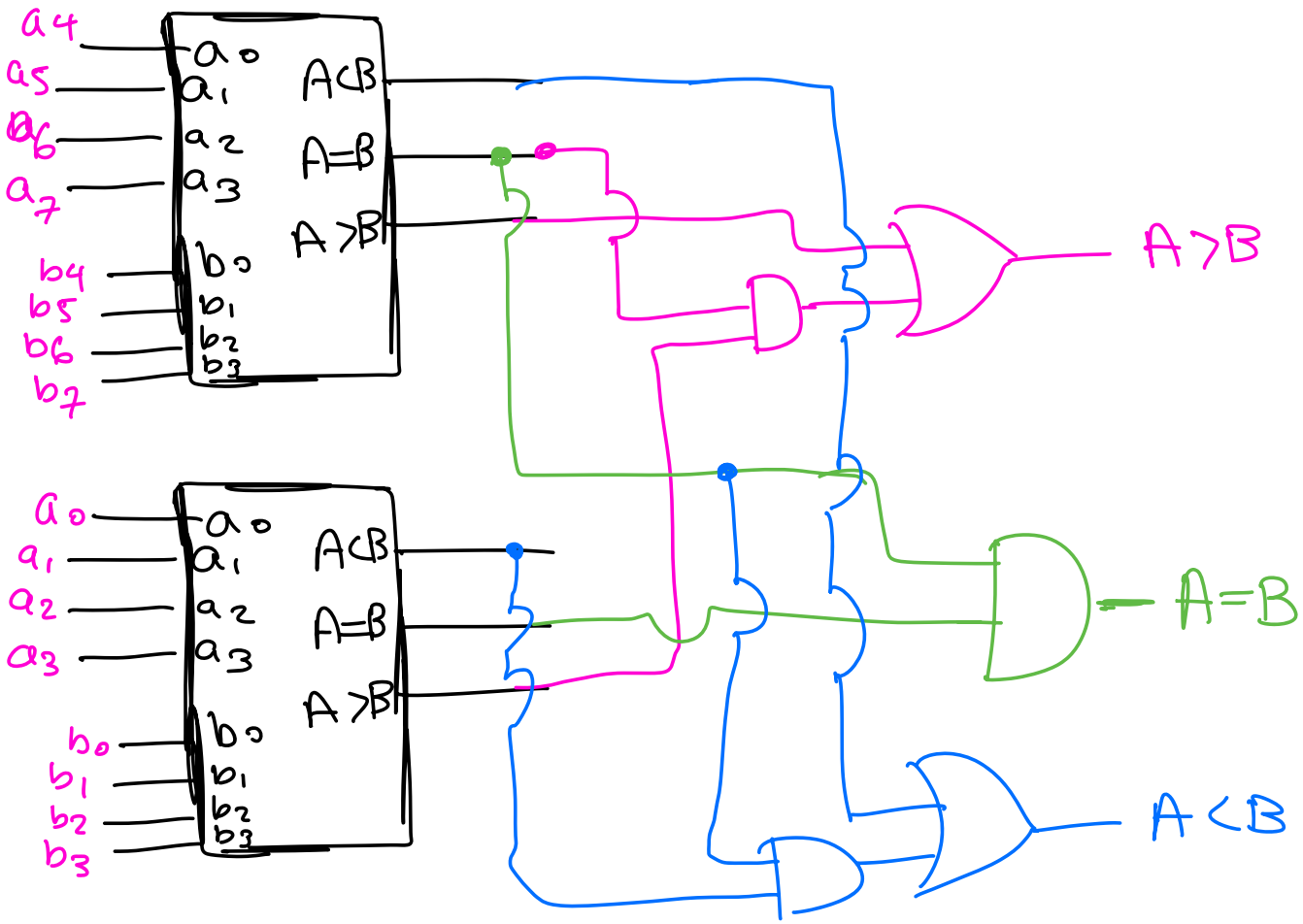


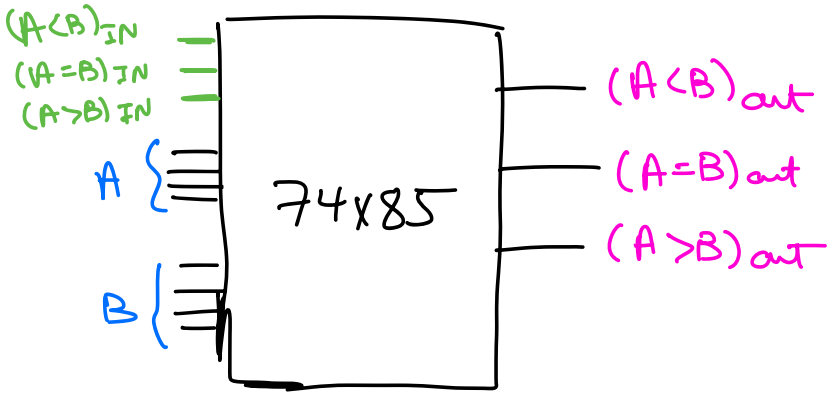
4-bit magnitude Comparator

Build an 8-bit magnitude comparator using two 4-bit comparators

A: $a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0$

B: $b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$

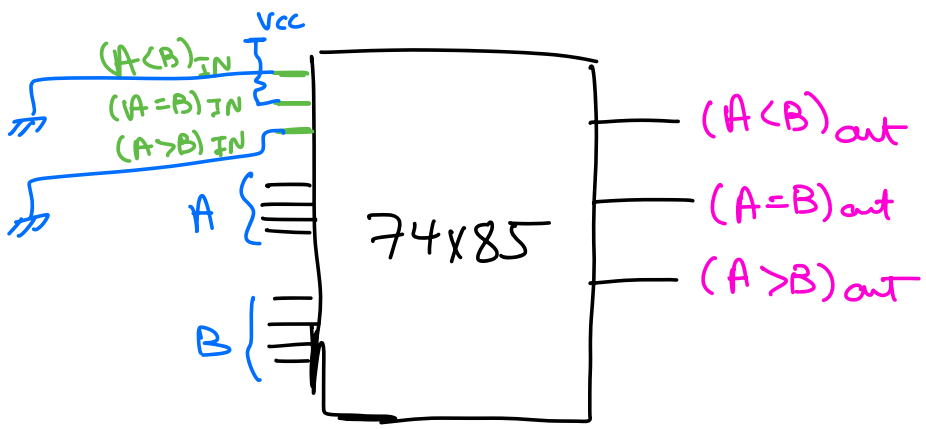




$(A=B)_{out} = (A=B) \cdot (A=B)_{IN}$

$(A>B)_{out} = (A>B) + (A=B) \cdot (A>B)_{IN}$

$(A<B)_{out} = (A<B) + (A=B) \cdot (A<B)_{IN}$

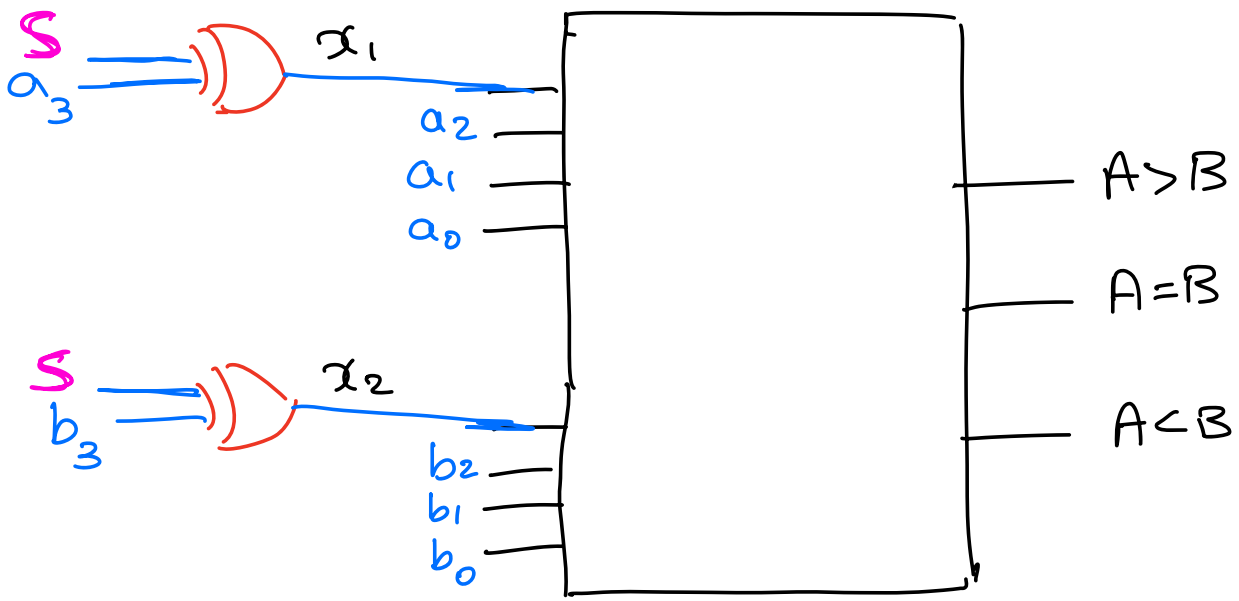


↑ How to use 74X85 as a 4-bit Comparator

Practice: use 74X85 to build a 12-bit Comparator.

A: a_3 a_2 a_1 a_0

B: b_3 b_2 b_1 b_0



when S=0 : unsigned numbers Comparator

when S=1 : 2's-Complement numbers Comparator

S: Control Signal → S=0 unsigned numbers
→ S=1 2's-Complement numbers

when S=0 $\begin{cases} x_1 = a_3 \\ x_2 = b_3 \end{cases}$

$x_1 = S.a_3' + S'.a_3$

$x_2 = S.b_3' + S'.b_3$

Typical operation of magnitude Comparators for unsigned numbers.

when $S=1$: $\begin{cases} x_1 = a_3' \\ x_2 = b_3' \end{cases}$

a_3	b_3		x_1	x_2
0	0	$A>0$, $B>0$	1	1
1	1	$A<0$, $B<0$	0	0
0	1	$A>0$, $B<0$	1	0

Example: $\begin{cases} A = 0111 & (+7) \\ B = 0110 & (+6) \end{cases}$ $\xrightarrow{S=1}$ $\begin{matrix} 1111 \\ 1110 \end{matrix}$ $A>B$

$(-1)\times 2^3 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0 = -1$

Example : $\begin{cases} A = 1111 & (-1) \\ B = 1110 & (-2) \end{cases}$ $\xrightarrow{S=1}$ $\begin{matrix} 0111 \\ 0110 \end{matrix}$ $A>B$

Example : $\begin{cases} A = 0111 & (+7) \\ B = 1110 & (-2) \end{cases}$ $\xrightarrow{S=1}$ $\begin{matrix} 1111 \\ 0110 \end{matrix}$ $A>B$