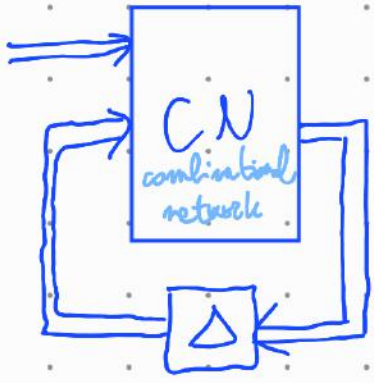
POSSIBLE FUNCTIONS WE CAN BUILD

Combinational → output depends on actual input

Sequential → output depends on actual input and previous outputs (memory)

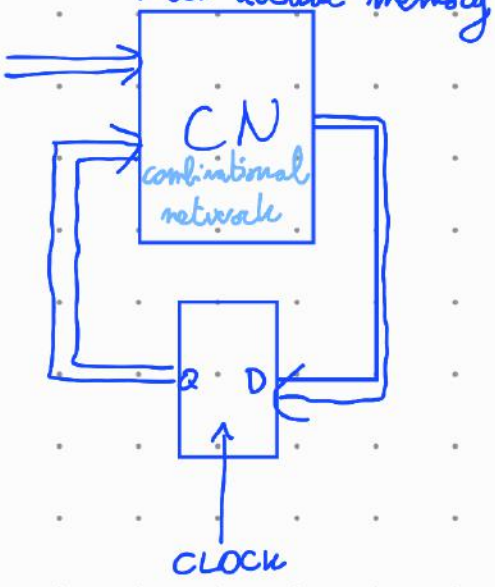
ASYNCHRONOUS

With Delay



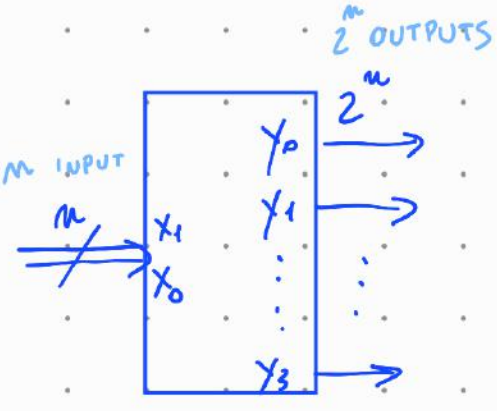
SYNCHRONOUS

With actual memory



BINARY DECODER

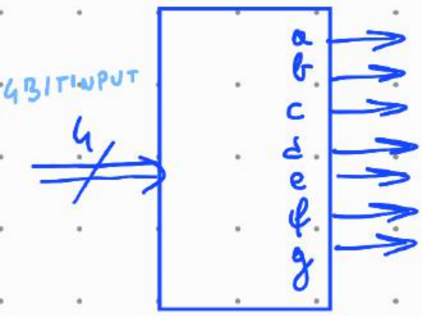
converts an INPUT to a particular OUTPUT



x_1	x_0	y_0	y_1	y_2	y_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

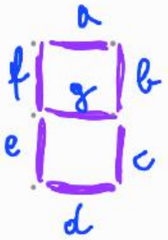
Activates only the corresponding value from the current input.

DECODER FOR 7-SEGMENT DISPLAY



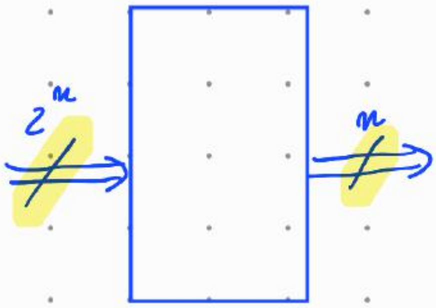
X_3	X_2	X_1	X_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
⋮							⋮			

7 segment display



BINARY ENCODER

Converts a LARGER INPUT to a SMALLER OUTPUT



PRIORITY ENCODER TABLE (EX.)

X_3	X_2	X_1	X_0	y_1	y_0
1	x	x	x	1	1
0	1	x	x	1	0
0	0	1	x	0	1
0	0	0	1	0	0

Gets the only input activated and returns the corresponding code.

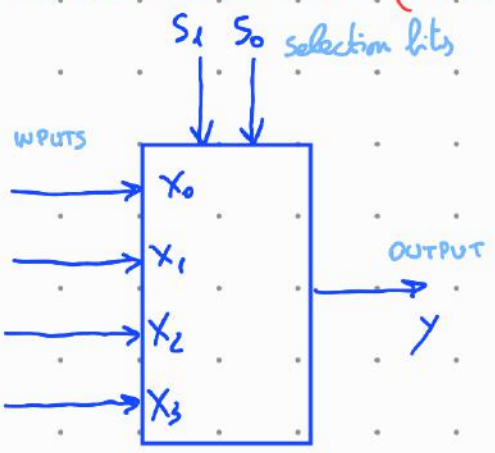
INPUT COMBINATIONS: 2^{2^n} not all are valid! Only the ones with only one "1" are valid.

$(2^{2^n} - 2^n)$ INVALID codes

PRIORITY ENCODER

Allows multiple activated lines. Returns the higher priority one.
It is used to manage peripherals.

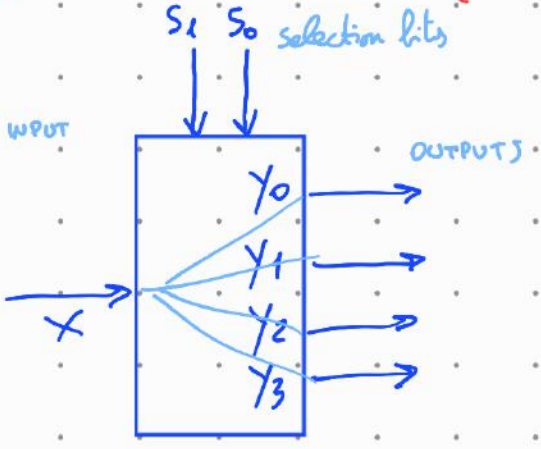
MULTIPLEXER (MUX)



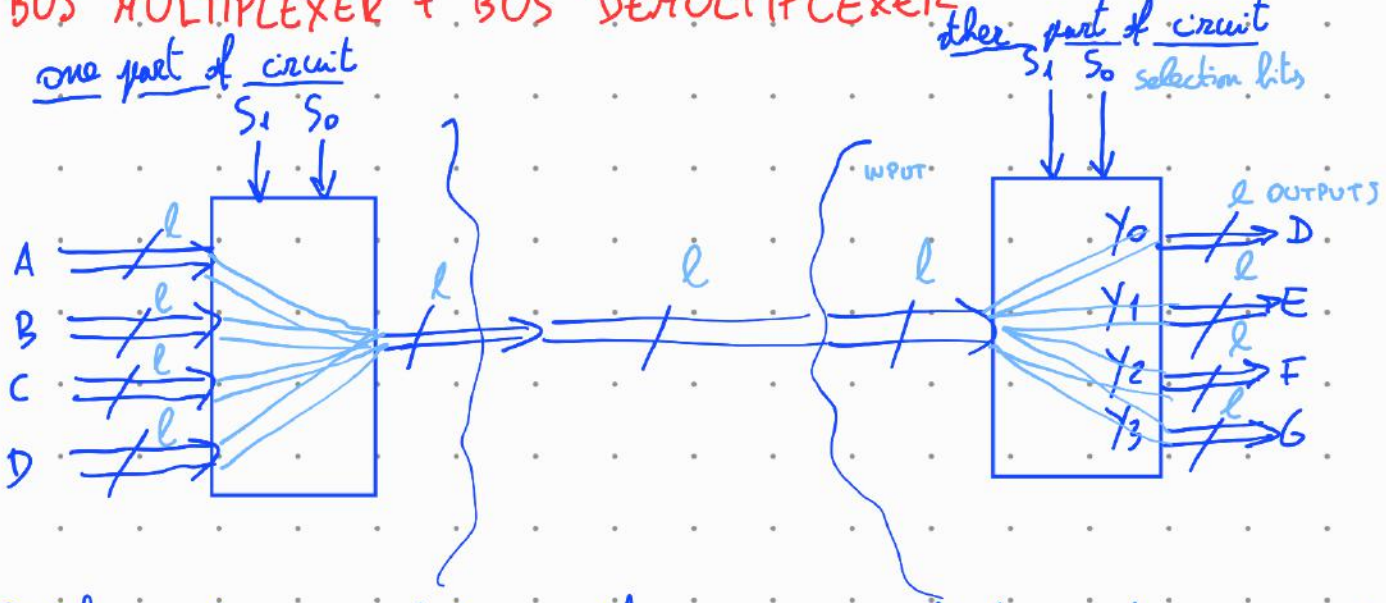
S_1	S_0	Y
0	0	X_0
0	1	X_1
1	0	X_2
1	1	X_3

Selects which data I want to deliver as output. We must have $\log_2(\#\text{INPUTS})$ selection bits.

DE MULTIPLEXER (DEMUX)



BUS MULTIPLEXER + BUS DEMULTIPLEXER



Both ends must know the sequence of channel used to talk.

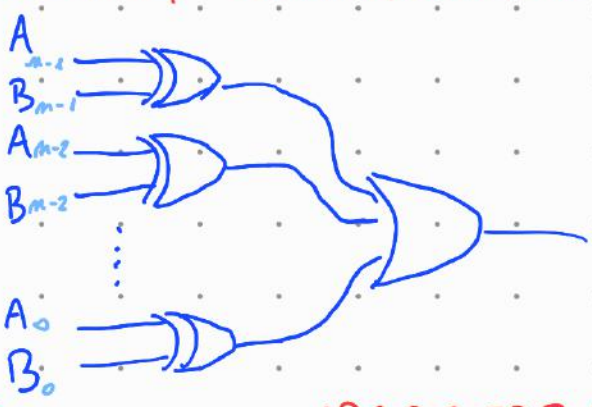
XOR (exclusive OR)

$Y = A\bar{B} + \bar{A}B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Can be used as a (dis)equality comparator.

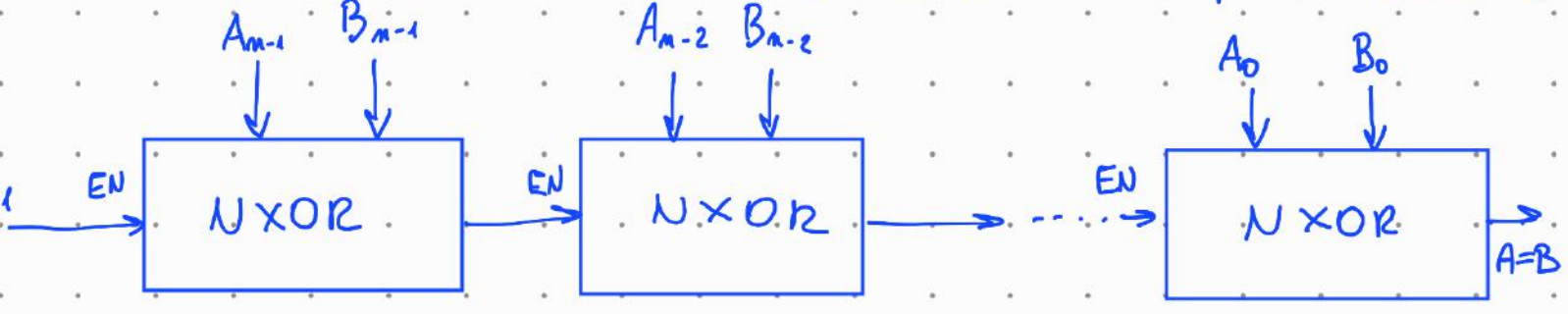
BINARY COMPARATOR 1



OR needed to say output, only one needs to be true

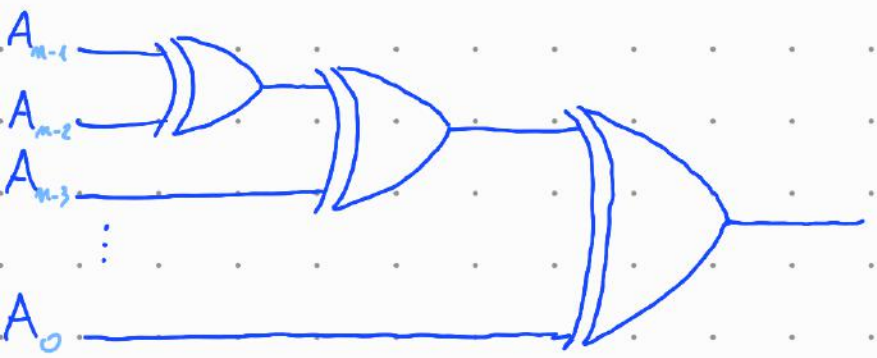
BINARY COMPARATOR 2

We start comparing from the most significant bit,
NXOR (return "1" if equal, "0" otherwise)



If we conclude with a 1 on the last block, than the two numbers are equal. If we do not arrive at the end, it means that one bit is different, so the numbers are not equal.

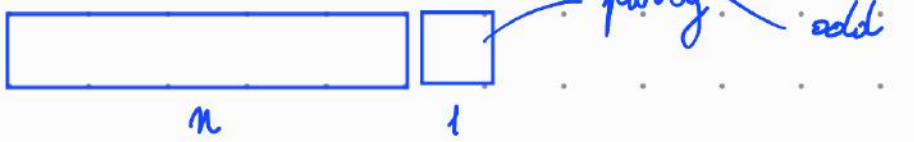
XOR CASCADE



Returns "1" if the number A has an odd (dispari) number of "1", "0" otherwise.

Can be useful to calculate parity bit of a number.

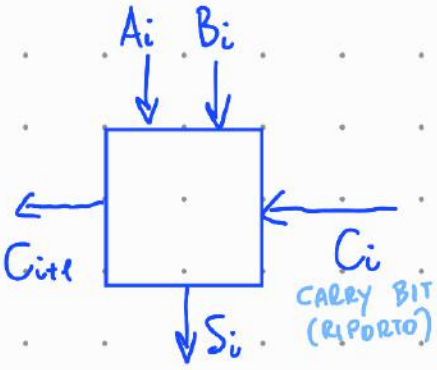
PARITY CHECK



It is useful as TRASMISSION CONTROL technology. If the receiver gets odd-number of bit-flip number than the parity bit will be incorrect and request a retrasmision to the sender.

More sofisticated versions can even fix the bit-flip phenomenon, but require more information.

FULL ADDER



Full-adder

RIPORTO ATTUALE (DA CONSIDERARE ORA)
ELEMENTI DA SOMMARE
RISULTATO SOMMA ATTUALE

Ci	Ai	Bi	Si	Ci+1
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	1	1	1

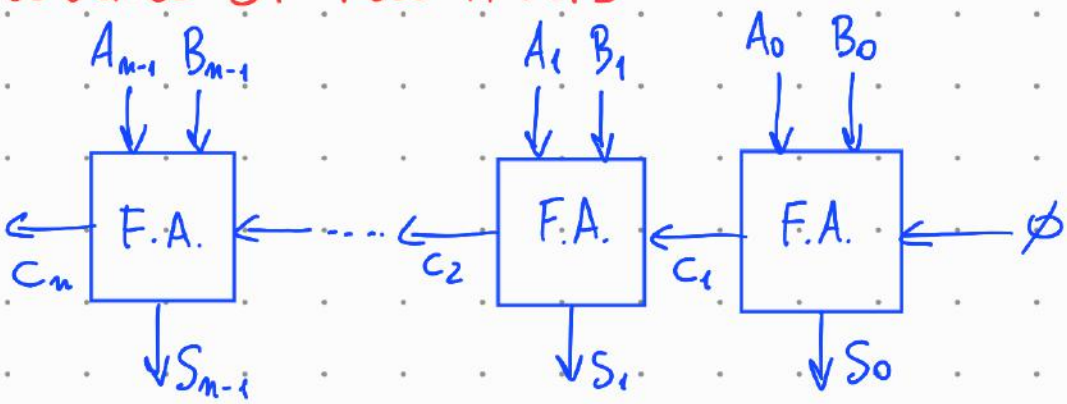
SE GIÀ 1 E VA SOMMATO "1" METTI A "0" E SEGNA IL RIPORTO DA PASSARE A QUELLO DOPO

$$S_i = A_i \oplus B_i \oplus C_i$$

"Si" counts the number of "1", made of XOR

$$C_{i+1} = (A_i \cdot B_i) + C_i$$

SEQUENCE OF FULL-ADDERS



NEGATIVE NUMBERS

2 methods:

1) Sign bit

SIGN BIT — NUMBER BITS

NOT SO USED

2) 2's Complement

sign

$$A = \{A_{n-1}, A_{n-2}, \dots, A_1, A_0\}$$

↑ represents also the sign

$$A = -A_{n-1} \cdot 2^{n-1} + A_{n-2} \cdot 2^{n-2} + \dots + A_1 \cdot 2 + A_0 \cdot \underbrace{2^0}_{=1}$$

$-A_{n-1} \cdot 2^{n-1}$: represents the sign only

$A_{n-2} \cdot 2^{n-2} + \dots + A_1 \cdot 2 + A_0 \cdot 2^0$: not the complement of the number

With this 2's complement we can use the sequence of full-adders without changing the structure (except for the first F.A.) for the subtraction.

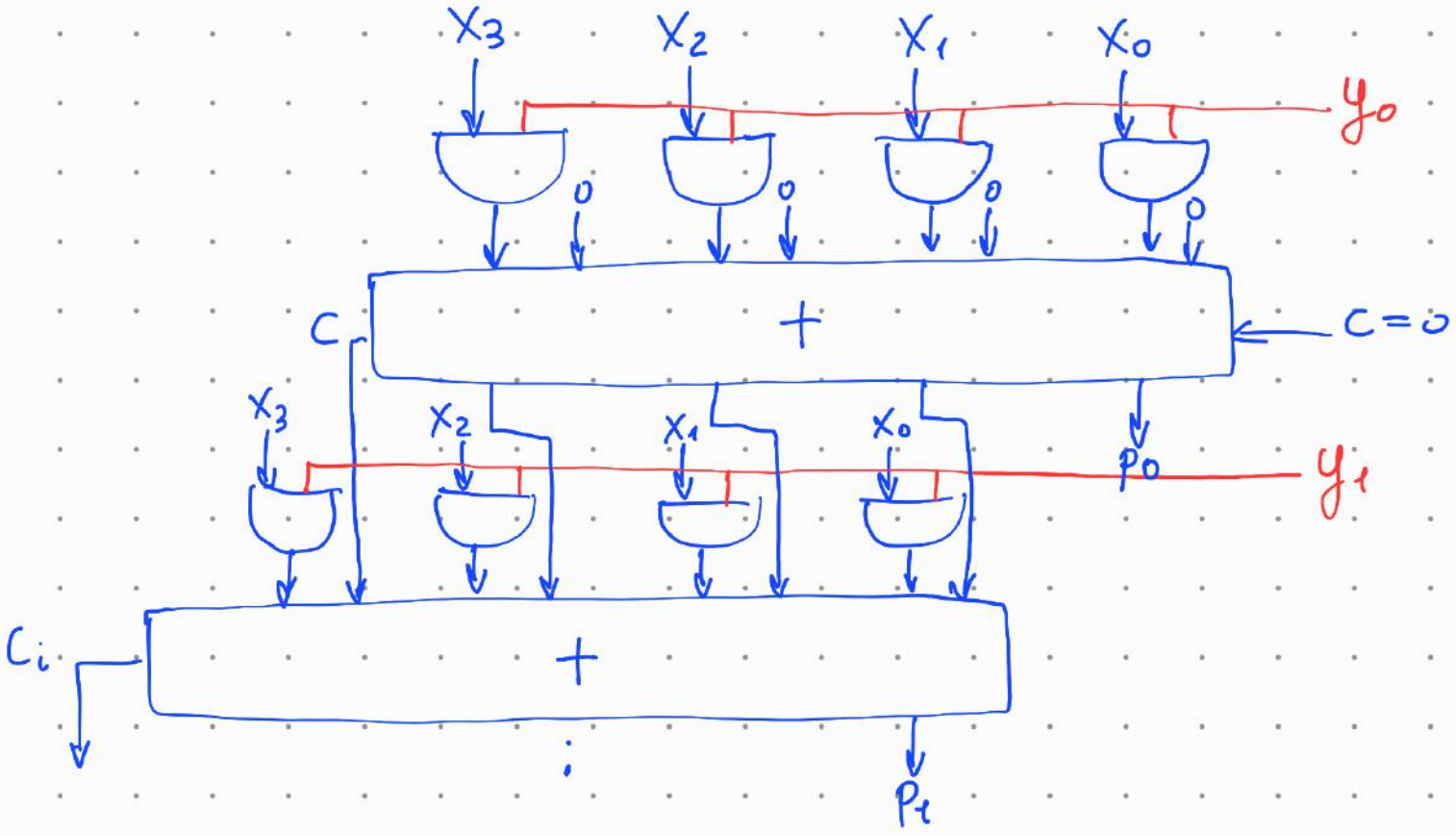
SHIFT AND ADD ALGORITHM FOR MULTIPLICATION

moltiplicando 1000
moltiplicatore 1001 =

moltiplicazione bit a bit solo con porte "AND"

1000
0000
0000
1000
1001000 risultato

MULTIPLICATION



$(X_3\ X_2\ X_1\ X_0) \cdot y_0$ sono aggiunti al prodotto di y_1 e $(X_3\ X_2\ X_1\ X_0)$ spostato a sinistra.

La DIVISIONE è più complicata e richiede più di 1 clock.

COMPLEMENTO A DUE

Per rappresentare "−5":

– scrivo "5"	0 0 0 0 0 1 0 1
– inverto tutto	1 1 1 1 1 0 1 0
– sommo 1	1 1 1 1 1 0 1 1

il risultato è "−5" in complemento a due, so che è negativo perché il primo bit a sin. è 1

per sapere il numero, devo fare al contrario:

– inverto tutto	0 0 0 0 0 1 0 0
– sommo 1	0 0 0 0 0 1 0 1

so che il numero negativo è "5".

(+)5 in complemento a due è sempre 0101, si capisce dal 1° bit che è positivo.