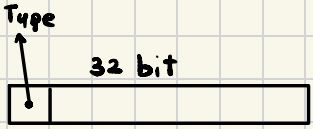
Livelli di astrazione dei sistemi di calcolo

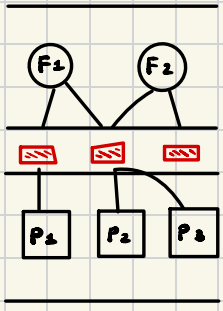
Applicazioni
Sistema Operativo
Architettura
Componenti logiche
Dispositivi fisici

→ "ArmV7", "x86_64"

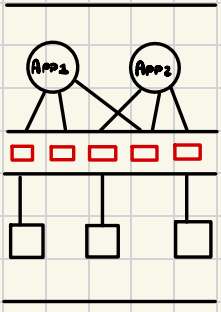


Differente tra "meccanismi" e "politiche".

- Livello i
- Meccanismi
- Livello i-1



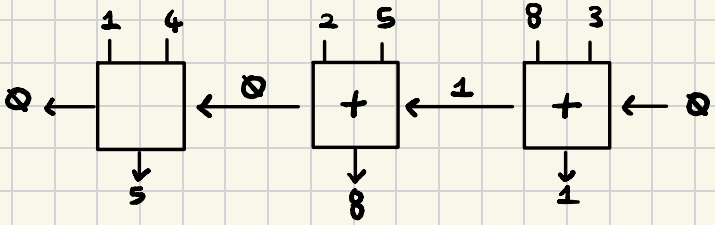
=>



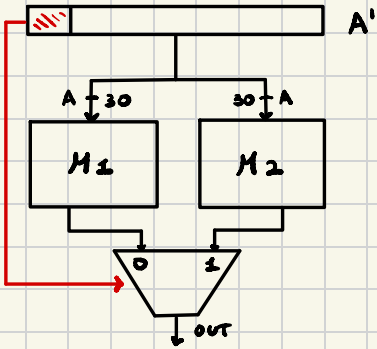
- Applicazioni
- Sistema operativo

Principi di progettazione

(1) Principio di gerarchia, progettare una componente complessa in termini di componenti più piccole, di cui è facile comprenderne il funzionamento e siano facili da progettare.

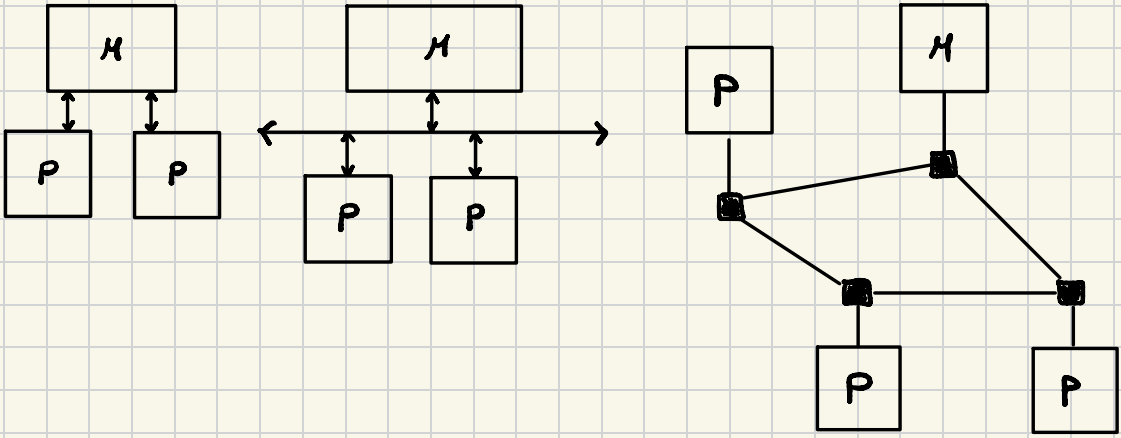


(2) Principio di **modularità**, il sistema deve essere composto da componenti / moduli che implementano funzionalità ben chiare e di chiara semantica, e con interfacce che favoriscano la composizione di più componenti per realizzare sistemi complessi.

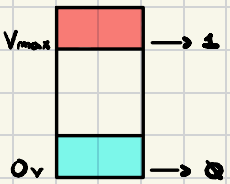


- Ogni modulo gestisce parte dell'input.
- Collegando i moduli hai > capacità.
- Il multiplexer decide che modulo usare.

(3) Principio di **regolarità**, una buona progettazione implica l'identificazione di funzionalità e moduli comuni, che possono essere riutilizzati per implementare funzionalità diverse riducendo il numero di componenti base necessarie alla progettazione complessiva.



Sistema Binario



- $728_{10} = (8 \cdot 10^0) + (2 \cdot 10^1) + (7 \cdot 10^2)$
- $1011_2 = (0 \cdot 2^0) + (1 \cdot 2^1) + (1 \cdot 2^2) + (0 \cdot 2^3) + (1 \cdot 2^4) = 22_{10}$

Conversione Decimale -> Binario

84	2
42	0
21	0
10	1
5	0
2	1
1	0
	1

- $1010100 = 84_{10}$

Metodo alternativo, togliere la potenza di 2 >.

- $84 - 64 = 20 - 16 = 4 - 4 = 0$
  $2^6 + 2^4 + 2^2 = 84.$

Sistema Esadecimale

- $\Gamma = \{0,1,2 \dots 9, A, B, C, D, E, F\}$
- 0x indica la base 16.
- $0x2ED = 0010\ 1110\ 1101$
  (4 4 4) → 2 E D

Somma tra binari negativi

$5_{10} = 000101$

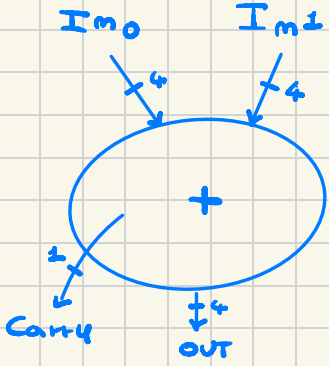
$18_{10} = 010010$

0 0 0 1 0 1	+
0 1 0 0 1 0	=
0 1 0 1 1 1	$= 23_{10}$

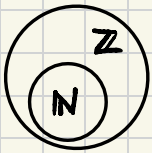
BIT CARRY

* (1) + 15

1 1 1	
1 (carry)	
0 0 0 1	+
1 1 1 1	=
0 0 0 0	



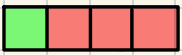
Rappresentazione numeri relativi



$\mathbb{Z}$
- Modulo e segno. (1)
- Complemento a2. (2)

(1) Modulo e segno

- m = 4 bit



m bit → $[-2^{m-1}+1 , 2^{m-1}-1]$

! **-1 + 2 = 1**

```
1 0 1 +
0 1 0
-------
1 1 1  = -3 !
```
→ Non torna!

Complemento a 2

- Si cambiano i numeri: 0 → 1 , 1 → 0 , poi si +1.

Esempio:

- 7 = 0111 => -7 = **1000 + 1 = 1001**

- 5 + (-7) = -2 =

```
0 1 0 1 +
1 0 0 1 =
---------
1 1 1 0  = -2
```

```
1110 ⇝ 0001 +
             1
-------------
0010 = 2.
```

OPERAZIONI IN COMPLEMENTO A2

24/09/2025

m = 4 • -5 → 0101 → 1010 + 1 = 1011 = -5

ESEMPIO: m = 6 bit

• 14 + 9

$14 = 2^3 + 2^2 + 2^1 = 001110 +$

$9 = 2^3 + 2^0 = 001001 =$

$010111 = 23$

ESEMPIO 2:

• 17 - 13 =

$17 = 2^4 + 2^0 = 010001 +$

$-13 = 110011 =$

$000100 = 4$

ESEMPIO 3:

• 25 + 18

25 = 011001 +

18 = 010010 =

101011

• Se il riporto sulla colonna significativa e quello in uscita sono diversi → Overflow

ALGEBRA BOOLEANA

$\Gamma = \{0, 1\}$ NOT, AND, OR

NOT → Negazione logica $\bar{A}$

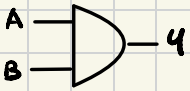
AND → Congiunzione logica *

OR → Disgiunzione logica +

PORTA NOT:

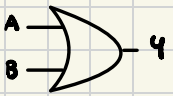
A	Y
0	1
1	0

PORTA AND:



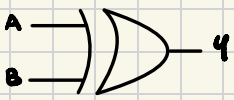
A	B	Y
0	0	0
1	0	0
0	1	0
1	1	1

PORTA OR:



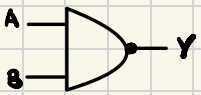
A	B	Y
0	0	0
1	0	1
0	1	1
1	1	1

PORTA XOR:



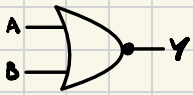
A	B	Y
0	0	0
1	0	1
0	1	1
1	1	0

PORTA NAND:



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

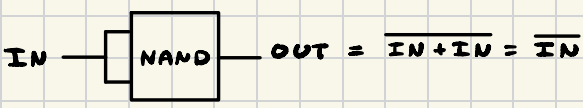
PORTA NOR:



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

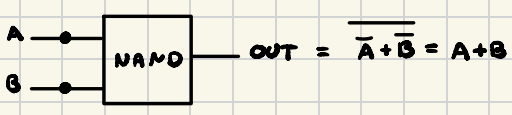
TECNOLOGIA CMOS:

NOT:

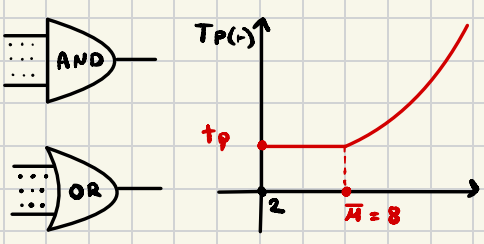


AND:

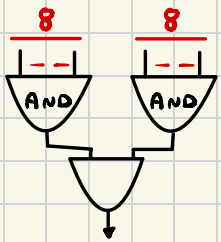
OR:



RITARDI DI PROPAGAZIONE DELLE PORTE LOGICHE:



- Minimo tempo per avere la risposta.

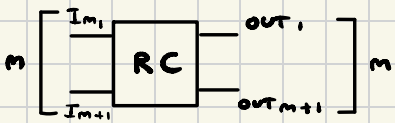


- Ritardo totale = 2TP

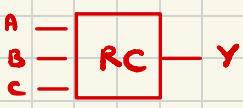
COSTO TOTALE: $\lceil \log_8 m \rceil \cdot tp$

RETI COMBINATORIE

- RETI LOGICHE
  - COMBINATORIE. (NO MEMORIA)
  - SEQUENZIALI. (IMPLEMENTA UN'AUTOMA, MEMORIA)



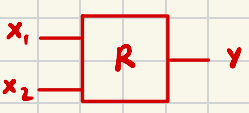
- $\forall i = 0 \dots m$ OUT =



$A, \bar{A}, B, \bar{B}, C, \bar{C}$ → Letterali

MINTERMINE ⇝ $A\bar{B}C$, prodotto logico di tutte le variabili

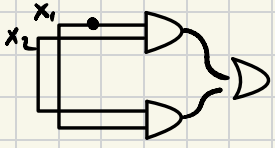
FORMA CANONICA IN "SOMMA DI PRODOTTI" (SP)



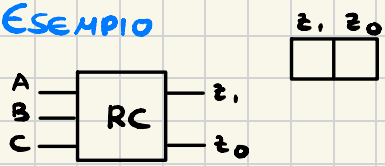
	x_1	x_2	y
$\bar{x}_1 \bar{x}_2$	0	0	0
$\bar{x}_1 x_2$	0	1	1
$x_1 x^2$	1	0	0
$x_1 x_2$	1	1	1

→ Devo trovare un'espressione booleana equivalente

$y = \bar{x}_1 x_2 + x_1 x_2$



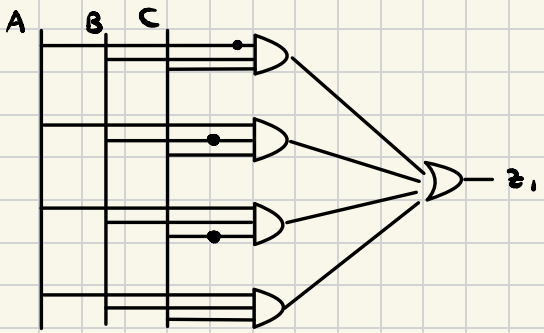
Esempio



A	B	C	z_1	z_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$z_0 = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$z_1 = A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$$

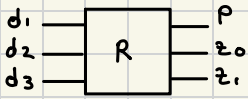


Esempio 2

...

Esempio con indifferenze in uscita

26/09/2025



d_1	d_2	d_3	P	z_1	z_0
0	0	0	0	-	-
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1

- $z_0 = \dots$

Proprietà dell'algebra booleana

$\bar{0} = 1$, $\bar{1} = 0$ Postolati

$A \cdot 0 = 0 \quad \forall A$

$A + 1 = 1 \quad \forall A$

$A + \bar{A} = 1 \quad \forall A$ Complemento

$A \cdot \bar{A} = 1 \quad \forall A$

$A \cdot B = B \cdot A$

$A + B = B + A$ → Commutativa

$A(BC) = (AB)C$

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

- $(B \cdot C) + (B \cdot D) = B(C + D)$ → Distributiva di $\wedge$ su $\vee$
- $(B + C)(B + D) = B + (C \cdot D)$ → " somma su prodotto

- $\overline{A + B} = \bar{A} \cdot \bar{B}$ De Morgan

Esempi

(1) $z = \bar{A}\bar{B}\bar{C}D + A\bar{B}\bar{C}D =$

$z = \bar{A}(\bar{B}\bar{C}D) + A(\bar{B}\bar{C}D) =$

$z = (\bar{B}\bar{C}D) \cdot \underbrace{(A + \bar{A})}_{1} = \bar{B}\bar{C}D$

(2) $z = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$

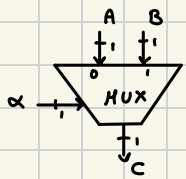
$z = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + A\bar{B}\bar{C}$

$z = \bar{B}\bar{C}\underbrace{(\bar{A} + A)}_{1} + A\bar{B}\underbrace{(C + \bar{C})}_{1} = \bar{B}\bar{C} + A\bar{B}$

- Im or mom cambia!

Esempio del "multiplexer"

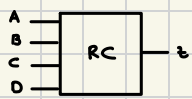
A	B	α	C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



- Se $\alpha = 0$ manda a in uscita, se $\alpha = 1$ manda B.

$$C = \bar{A}B\alpha + A\bar{B}\bar{\alpha} + AB\bar{\alpha} + AB\alpha$$
$$= B\alpha(\bar{A}+A) + A\bar{\alpha}(\bar{B}+B) = B\alpha + A\bar{\alpha}$$
($\bar{A}+A = 1$, $\bar{B}+B = 1$)

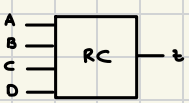
Mappe di Karnaugh (Mappek) → Approccio grafico per espressioni booleane.



$\Rightarrow$

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

- $A\bar{B}C\bar{D} + A\bar{B}CD = A\bar{B}C(\bar{D}+D) = A\bar{B}C$ ($\bar{D}+D = 1$)



$\Rightarrow$

CD \ AB	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	0	1	0
10	0	0	1	0

- Si cercano "cerchi" regolari, ovvero potenze di 2. => "rettangoli.
- Vanno cerchiati tutti gli 1 usando meno "cerchi" possibili.

- Se il "cerchio" è grande 2^k, si eliminano k variabili.
- Posso cerchiare numeri già cerchiati.

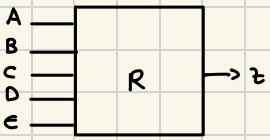
=> z = AB | $A\bar{C}$ | $\bar{A}\bar{B}CD$

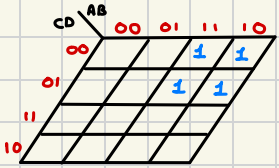
- Guardo chi mantiene il segno.

$$\square = AB\bar{C}\bar{D} + AB\bar{C}D + ABCD + ABC\bar{D}$$

$$= AB\bar{C}(\bar{D}+D) + ABC(D+\bar{D}) = AB\bar{C} + ABC = AB(\bar{C}+C) = AB$$

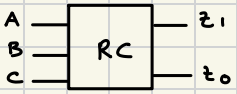
Limiti delle mappe k: Se ci sono troppi imgressi, si creamo problemi.





- Com + 4 imgressi vieme complicato, mom ci rigarda.

Contatori di 1ni su 3 bit

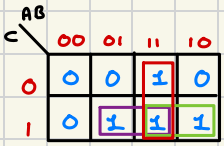


- $z_1 = \bar{A}BC + A\bar{B}C + A\bar{B}C + ABC$
- $t_0 = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

A	B	C	z_1	t_0
0	0	0	0	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	0
1	1	0	1	0
1	1	1	1	1

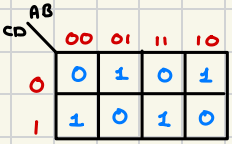
Mappe

$z_1 =$



$z_1 = BC + AB + AC$

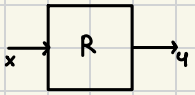
$t_0 =$



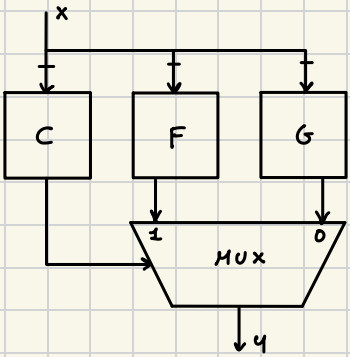
- 4 "cerchi" da 1, mom é amcora riducibile.

Approccio "strutturale" al progetto di reti combinatorie

01/10/2025



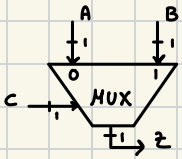
- If c(x) then $y = f(x)$;
  else $y = g(x)$;



- Il blocco C indica la funzione c(x) di controllo.
- Il blocco F calcola f(x).
- Il blocco G calcola g(x).
- Il MUX decide l'uscita =>
  - Se c(x) = 1 → F
  - Se c(x) = 0 → G

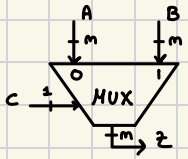
Componenti standard di "logica combinatoria"

1.1) Multiplexer base (MUX)



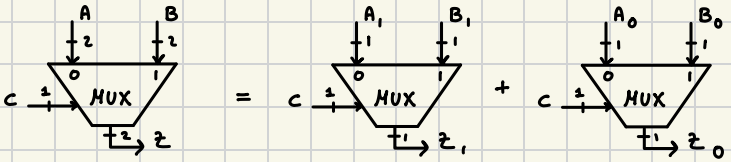
- Ingressi ad 1 solo bit (compreso il controllo).

1.2) Multiplexer con due ingressi a N>1 bit



- Ingressi con m bit, controintuitivamente in questo caso non c'è ritardo.

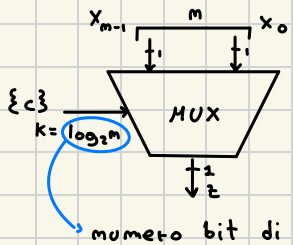
Esempio



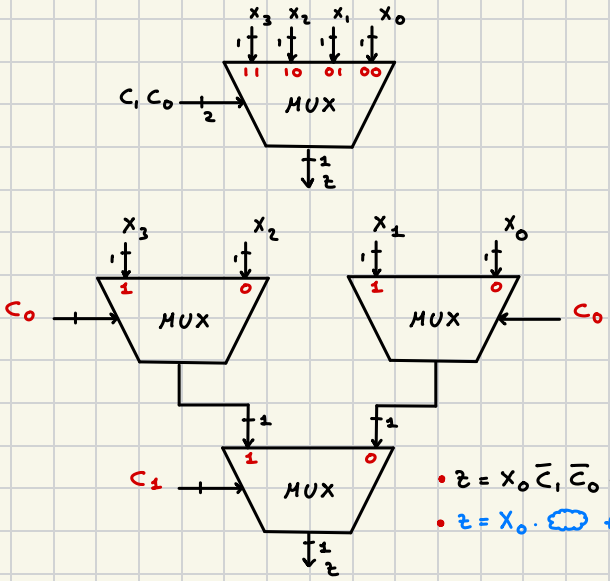
A_1	A_0	B_1	B_0	C	z_1	z_0
0	0	-	-	0	0	0
0	1	-	-	0	0	1
1	0	-	-	0	1	0
1	1	-	-	0	1	1
-	-	0	0	1	0	0
-	-	0	1	1	0	1
-	-	1	0	1	1	0
-	-	1	1	1	1	1

- Agiscono in parallelo (contemporaneamente).

1.3) Multiplexer con N ingressi a 1 bit



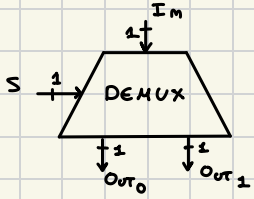
Esempio



X_3	X_2	X_1	X_0	C_1	C_0	t
–	–	–	0	0	0	0
–	–	–	1	0	0	1
–	–	0	–	0	1	0
–	–	1	–	0	1	1
–	0	–	–	1	0	0
–	1	–	–	1	0	1
0	–	–	–	1	1	0
1	–	–	–	1	1	1

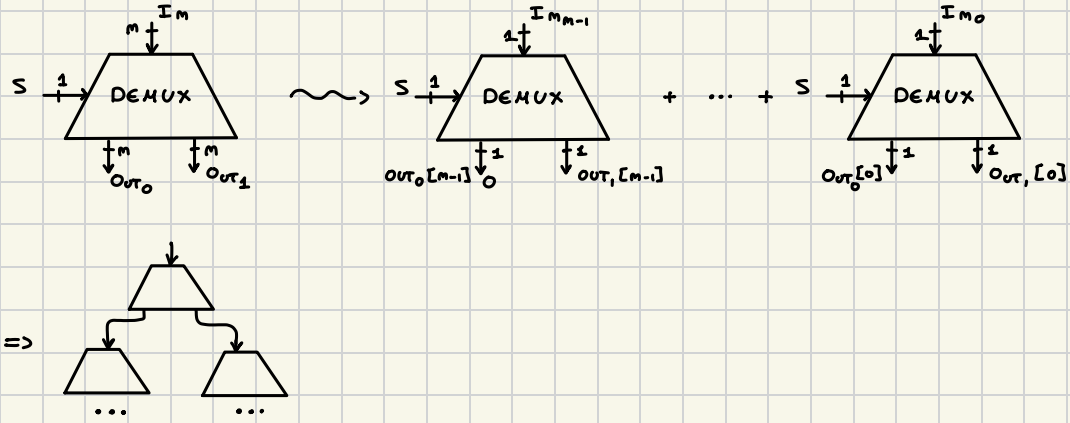
- $t = X_0 \bar{C}_1 \bar{C}_0 + X_1 \bar{C}_1 C_0 + X_2 C_1 \bar{C}_0 + X_3 C_1 C_0$
- $t = X_0 \cdot \text{☁} + X_1 \cdot \text{☁} + \ldots + X_{m-1} \cdot \text{☁}$

2) Demultiplexer (DEMUX)

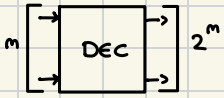


I_m	S	Out_1	Out_0
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0

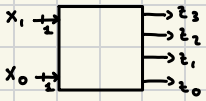
- $Out_0 = I_m \cdot \bar{S}$
- $Out_1 = I_m \cdot S$



3) DECODER (DEC)



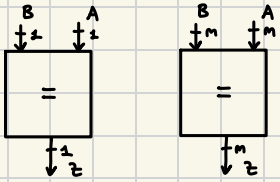
m = 2



X_1	X_2	z_3	z_2	z_1	z_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

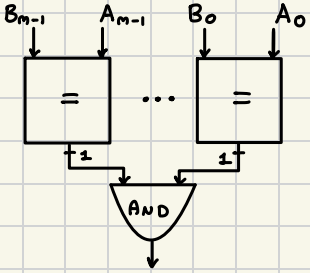
4) ENCODER (ENC) => Intuitivamente il contrario del decoder.

5) CONFRONTATORE



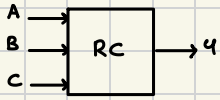
A	B	z
0	0	1
0	1	0
1	0	0
1	1	1

- $A = A_{m-1} \dots A_0$
- $B = B_{m-1} \dots B_0$



- I comfromtatori devomo essere messi im AND per capire se tutti i bit somo uguali.

MAPPE K CON INDIFFERENZE IN USCITA:



A	B	C	Y
0	0	0	-
0	0	1	1
0	1	0	-
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	-

C \ AB	00	01	10	11
0	-	-	0	1
1	1	0	-	1

- $y = \bar{A}\,\bar{B}\,C + A\,\bar{B}\,\bar{C} + A\,\bar{B}\,C$
- $y = \bar{B}$

- Indifferenza significa che alcuni out sono indefiniti, si può scegliere.
- Nelle mappek si deve scegliere in modo da minimizzare il n° cerchi e massimizzare la grandezza.