

Definizione delle funzioni

$$\text{es. } x = f(a)$$

$$x = f(a, b, c)$$

$\Rightarrow$  corrisponde a una

rappresentazione binaria, quindi i valori in uscita sono finiti.

INGRESSO	$x$	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
2 BIT	0	0	0	1	1
	1	0	1	0	1

possibili casi

"

corrispondono a 4 funzioni diverse

- $f_1$  da in uscita sempre zero
  - $f_4$  da in uscita sempre uno
- ] sono funzioni costanti (proprio chiamate "funzioni")

- $f_2, f_3$  invece dipendono dal valore di  $a$ .

$\rightarrow f_2$  è la funzione IDENTICA (dato un certo input mi ritorna lo stesso valore)

$\rightarrow f_3$  è la funzione di NEGAZIONE ("ritorna l'opposto")

Per ottenere più funzioni, dovrà aggiungere più variabili in ingresso: es.  $u = f(a, b)$

$u = f(a, b) \rightsquigarrow (a, b, c, d, \dots)$  date  $n$  variabili posso costruire  $2^n$  funzioni

con  $n=2$  ( $a \wedge b$ )

(l'esponente è costituito da  $n$  righe, che a loro volta assumono valore 0 o 1)

ho un totale di:  $2^2$

$= 2^4 = 16$  funzioni diverse

Sto costruendo delle TAVOLE DI VERITÀ

$a$	$b$	$u$	AND	altre così	OR	XOR	NAND	NOR	NXOR
0	0	0	0 0 0 0 1 0 0 1 0 0 1	0 0 0 0 1 0 0 1 0 0 1	0 0 0 0 1 0 0 1 0 0 1	0 0 0 0 1 0 0 1 0 0 1	1 1 1 1 0 1 1 0 1 1 0	1 1 1 1 0 1 1 0 1 1 0	1 1 1 1 0 1 1 0 1 1 0
0	1	0	0 0 0 0 1 0 0 1 1 1 1	0 0 0 0 1 0 0 1 1 1 1	0 0 0 0 1 0 0 1 1 1 1	0 0 0 0 1 0 0 1 1 1 1	1 0 0 0 0 1 1 1 1 0 0	1 0 0 0 0 1 1 1 1 0 0	1 0 0 0 0 1 1 1 1 0 0
1	0	0	0 0 0 0 1 0 0 1 0 1 1	0 0 0 0 1 0 0 1 0 1 1	0 0 0 0 1 0 0 1 0 1 1	0 0 0 0 1 0 0 1 0 1 1	1 1 1 1 0 1 1 0 1 0 0	1 1 1 1 0 1 1 0 1 0 0	1 1 1 1 0 1 1 0 1 0 0
1	1	0	0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 1

così di  $a \wedge b$

così = funzioni

Le funzioni costanti e d'identità sono poco interessanti, mentre altre restituiscono determinati valori, utile (valori esistenziali), citati di seguito:

- AND: 1 quando gli ingressi sono 1

- **AND**: 1 quando gli ingressi sono 1
- **OR**: 1 quando almeno uno degli ingressi è 1
- **XOR**: 1 quando solo uno dei due ingressi è 1 (detto "OR esclusivo")
- **NAND**: 1 sempre meno entrambi gli ingressi 1
- **NOR**: 1 solo quando entrambi gli ingressi 0
- **XNOR**: 1 quando gli ingressi hanno lo stesso valore di verità (0 o 1)

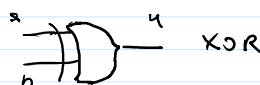
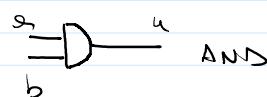
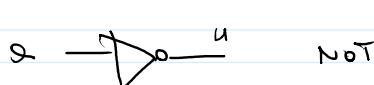
• Potrei procedere con 3 ingressi:  $f(a, b, c)$  ovvero  $z^3 = z^8$ : 256 funzioni diverse, con un notevole tasso di stesura.

• Si identificano dunque delle funzioni di base (**NOT**, **AND**, **OR**) e si costruiscono da queste tutte le altre.

### RAPPRESENTAZIONE DI FUNZIONI

Vi sono vari metodi per rappresentare le funzioni:

- 1) elencare i vari tipi per nome (NAND, XOR ecc... - vedi sopra)
- 2) con una **TAVOLA DI VERITÀ** in cui si confrontano valori in ingresso e uscita (vedi sopra)
- 3) rappresentazione grafica (circuiti):



Potrei concatenare i diversi operatori (simboli) e ottenere formule più complesse.  
(funzioni)

- 4) rappresentazione **algebrica booleana**

→ in ingresso uso le lettere a, b, c...

a	b	AND
0	0	0

a	b	OR
0	0	0

→ in ingresso uso le lettere a,b,c...

$$\text{NOT } \bar{a} \quad (a=0, \bar{a}=1)$$

(exOR) OR + SOMMA  
non usata

AND • Moltiplicazione

$a \ b$	AND	$a \ b$	OR
0 · 0	0	0 + 0	0
0 · 1	0	0 + 1	1
1 · 0	0	1 + 0	1
1 · 1	1	1 + 1	1

sono considerati  
giorni come un numero  
comprese fra zero e uno

$$0+0 = 0 \\ 0+1 = 1 \\ 1+0 = 1 \\ 1+1 = ?$$

e mi accorgo che il prodotto  
non dà il risultato

$$0 \cdot 0 = 0, \quad 0 \cdot 1 = 0, \\ 1 \cdot 0 = 0, \quad 1 \cdot 1 = 1$$

se noto 1 sarebbe  
contro la  
rappresentazione  
binaria

se noto 0 è letto,  
come se  
avesse  
overflow  
e non  
rappresenta 1

Sfruttando le proprietà algebraiche  
di queste funzioni, posso ovviare a  
tale problema

• COMUTATIVA:  $a+b = b+a$   
 $a \cdot b = b \cdot a$

• ASSOCIAZIONE, MISTRAZIONE... moltiplicazione delle AND e OR → posso così costruiremi varie diverse rappresentazioni algebrica

Penso rappresentare anche XOR,

anche se non presente nell'algebra logica  
(non ho i simboli binari)

→ ci occido usando i simboli binari

$a \ b$	XOR
0 · 0	0
0 · 1	1
1 · 0	1
1 · 1	0

ignoro le righe che escono da zero

Penso pensare alle somme di due prodotti

$$1^{\text{a}} \rightarrow \bar{a} \cdot b \quad \left. \begin{array}{l} \text{SOMMA} \\ \text{(OR)} \end{array} \right\} + \quad 2^{\text{a}} \rightarrow a \cdot \bar{b}$$

+ → ottengo la funzione u

Esempio

$a \ b \ c$	$u$
0 0 0	1 ← 1 <sup>a</sup>
0 0 1	0
0 1 0	0

invece sono f

• identifico le righe con  $u=1$

• cerco di ottenere le somme di tre prodotti  
(uno x ogni riga con 1 in esecuzione)

0 1 0	0
0 1 1	0
1 0 0	1 ← 2 <sup>a</sup>
1 0 1	1 ← 3 <sup>a</sup>
1 1 0	0
1 1 1	0

(uno x ogni argo con 1 in esclusione)

per sfruttare l'AND ( $\cdot$ ) devo farlo diventare 111, ovvero

$$u = \overline{a} \cdot \overline{b} \cdot \overline{c}^{\text{on}} + \overline{a} \cdot \overline{b} \cdot \overline{c}^{\text{on}} + \overline{a} \cdot b \cdot c^{\text{on}}$$

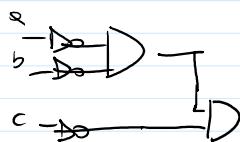
Ottengo  $u=1$  come <sup>dunque</sup> delle righe nere

non calcolo il risultato  
in modo esattamente

Penso tornare alle rappresentazioni grafiche:

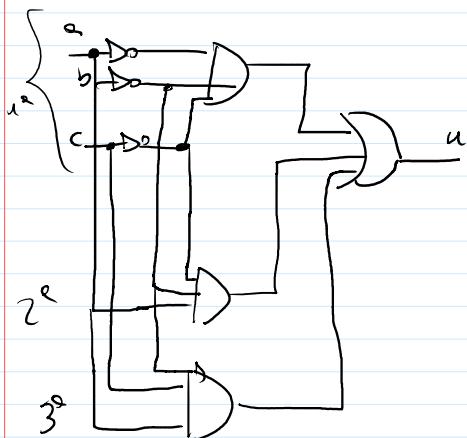


diventa



ovvero  $(\overline{a} \cdot \overline{b}) \cdot \overline{c}$  (per la 1<sup>a</sup>)

che è uguale a  $\overline{a} \cdot (\overline{b} \cdot \overline{c})$   
(e' commutativa)



Trasformazione

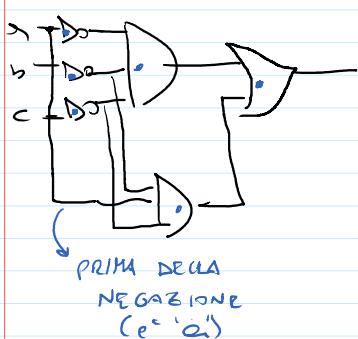
$$\cdot \overline{a} \cdot \overline{b} \cdot \overline{c} + a \cdot \overline{b} \cdot \overline{c} = (\underbrace{\overline{a} + a}_{=1}) \cdot \overline{b} \cdot \overline{c} \quad \text{ho scattato il termine comune } \overline{b} \cdot \overline{c}$$

$\downarrow$

1 - qualcosa = qualcosa quindi  $= \overline{b} \cdot \overline{c}$

1 - qualcosa = qualcosa quindi  $= \overline{b} \cdot \overline{c}$

$\downarrow$



6 oggetti



molto più semplice,  
SENZA SOMME

(e' 'a')

N.B.

Puoi ancora semplificare per le leggi di De Morgan

(ma non riguarda più l'algebra, bensì la logica)

b c u

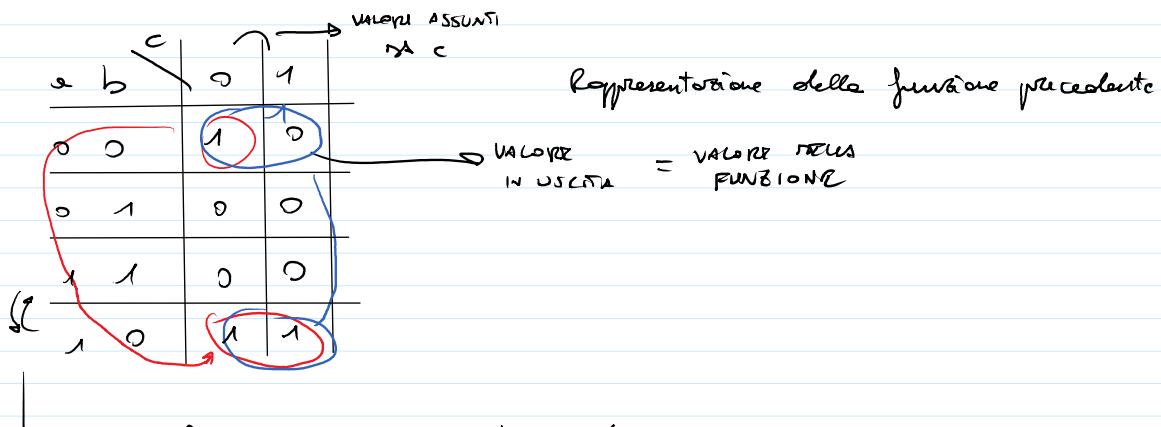
(b.c)  $\Rightarrow$  D<sub>0</sub>

0 0 1 e' il NOR  
0 1 0  
1 0 0  
1 1 0

Si è poi giunti a un procedimento rappresentabile con gli algoritmi, tramite una

5) rappresentazione alternativa, automatizzata: rappresentazione delle MAPPE DI KARNAUGH

→ rappresentazione bidimensionale (matrici bidimensionali)



→ Lo scambio può essere rilegato con le DISTANZA DI HAMMING:

1 0 0 1  
1 0 1 1 0

la distanza si calcola osservando un BIT alla volta:

REGOLE: stessi valori = 0 valori diversi = 1 e si sommano a livello numerico

$$\rightarrow H_d = 4$$

La distanza è definibile solo su stringhe di lunghezza eguale

11000 1101

111 0 0 1 0 0 1  
- - - - - 1 - -

$$H_d = 2$$

= n° BIT diversi (differiscono di 1 bit i due numeri)

La distanza fra due stringhe adiacenti ha valore 1, per questo lo SCARABIO

Lo differiscono di un solo BIT che permette di negligerlo, come

fatto precedentemente, anche queste altre regole di Karnaugh

es. se  $a=1$  e  $b=0$  non importa  $c$  ( $\Rightarrow 1$ ) perché ottengo sempre 1 e 1.

posso scrivere dunque  $u = a \cdot b$  versione semplificata di  $\overbrace{(c \cdot \bar{c}) \cdot a \cdot b}^{\text{(direttamente)}}$

Se considero la 1a riga  $\rightarrow \bar{a} \cdot \bar{b} \cdot \bar{c}$

mi emorgo che e' adiacente all'ultima, il che ci permette di annullare le caselle ( $\Rightarrow c=0$  e  $c=1$ )

Se due righe differiscono di  $a=0$  e  $a=1 \rightarrow (a \cdot \bar{a})$

e ottengo componendo  $u = \bar{a} \cdot b + \bar{b} \cdot \bar{c}$  versione già semplificata,

N.B.

Se invece di raggruppare coppie di caselle, posso raggruppare quattro, ottengo 1<sup>a</sup> e 4<sup>a</sup> righe che hanno b negato  $\rightarrow u = \bar{b}$  (versione ancora più semplificata)

Rappresentazioni viste:

✓ - TAVOLA DI VERITÀ

✓ - FORMULE ALGEBRAICHE

✓ - GRAFICA

✓ - MAPPE DI KARNAUGH

→ BDD  
BINARY DECISION DIAGRAMS

N.B.

Ognuna soddisfa certe esigenze

(efficienza, risorse ecc...)

→ sollesta x "scivolare a mani"

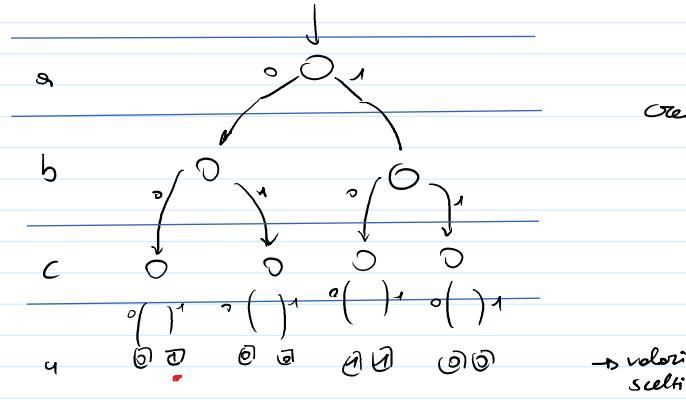
→ sollesta x usare degli algoritmi per semplificare le funzioni

ES 1

$f(a,b,c)$  prima di tutto bisogna ordinare le variabili  $a, b, c$ , che possono assumere valori 0 o 1  
ed ogni passo possa differenziare il valore zero e uno

ad esempio  
le elenco (non vi è un criterio x scegliere l'ordinamento struttura)

Root (origine)



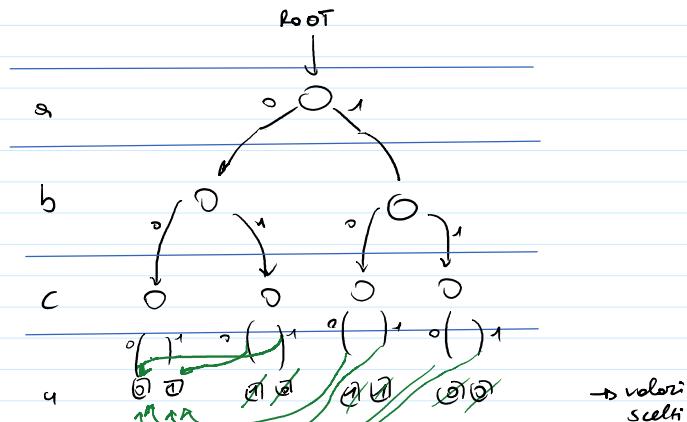
Crea una specie di ALBERO A LIVELLI

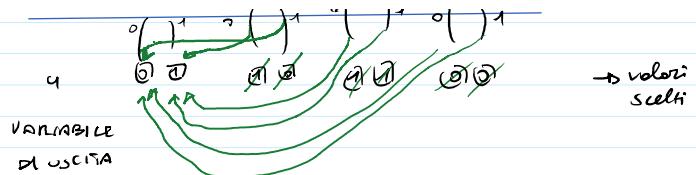
Se volessi trasformarla in una mappa di Karnaugh:

	b	c	0	1
0	0	0	0	1
1	0	1	1	0
dHam	1	1	1	0
1	1	0	1	0

Per economizzare posso riconoscere i punti equivalenti

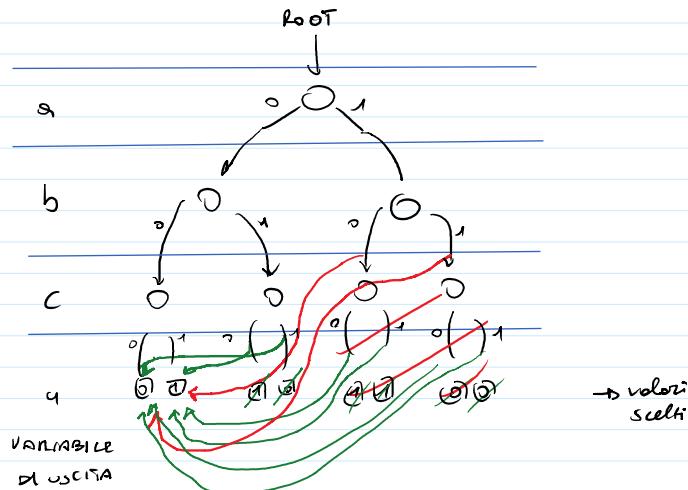
(ad esempio qui vediamo le foglie con valore zero)





Ho tolto delle foglie

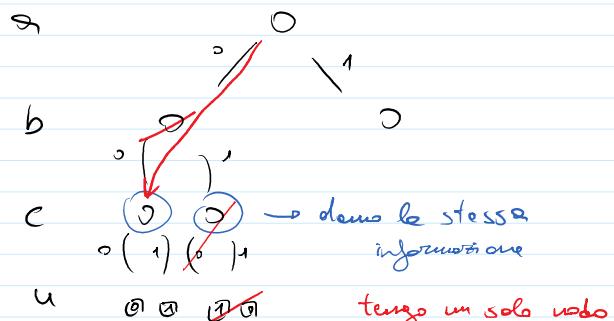
- osserviamo inoltre che sia quando  $c=0$  che  $c=1$  giungono nello stesso punto.



Ho tolto le foglie comuni

## Esercizio 2

	b	c	a	0	1
0	0	0		0	1
1	0	1		1	1
0	1	1		0	0
1				0	0
dove $\gamma = 1$					
$\begin{matrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{matrix}$					



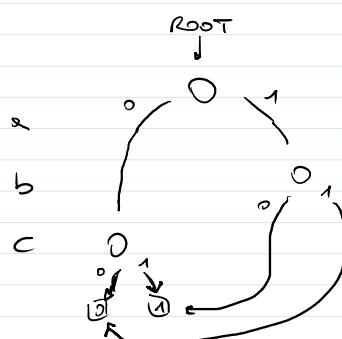
tengo un solo nodo

corrispondono  
all'unione  
di caselle  
adiacenti  
(posso semplificare  
le coppie nell'albero)



corrisponde a una semplificazione  
della MAPPA DI KARNAUGH  
in cui si rappresentano solo  
le coppie casellate

ottengo:



= con meno nodi e archi

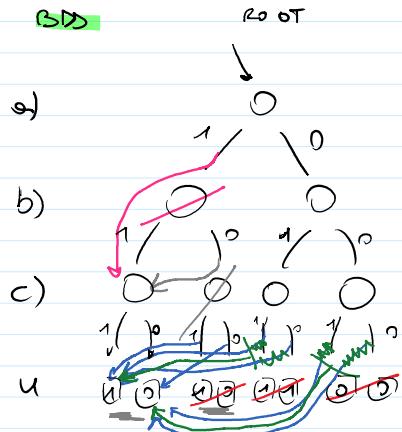
- stessa funzione
- 2 rappresentazioni diverse

Lo userò dove  
la funzione  
vale 1

per ottenere  $f(a,b,c) = 1$  (vera)

Ese corrisponde a  $\bar{a} \cdot c + a \cdot \bar{b}$  → prende  $a=1$ ,  $b$  vale 0 o 1  
quando  $\bar{a}=0$ , l' uscita  
è l' identità su  $c$

Es 3 (da appunti 2003) - elencatevi sotto le semplificazioni passo-passo.



MdK

a	b	c	0	1
0	0	0	0	0
0	1	0	1	1
1	1	0	0	1
1	0	0	0	1

) possibilità  
di semplificazione

- mantengo una sola foglia x uscita
- dirigo i nodi verso le foglie di sx
- sostituisco i doppi nodi con stesso valore in uscita
- quando c mi accorgo che due nodi calcolano la stessa cosa (valori in uscita uguali)
- per quanto detto nel gergo, non importa più se  $b=0$  o  $1$

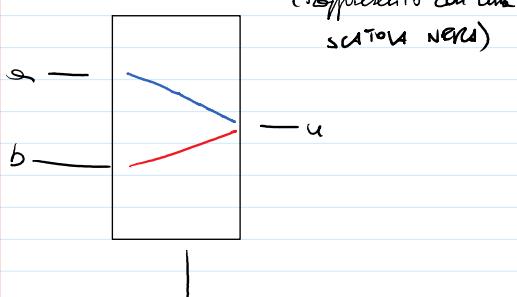
- le semplificazioni sono equivalenti  
alle due rappresentazioni
- Il modo in cui elenco le variabili  
nella BDD è indipendente da MdK  
(può anche essere  $bc \setminus a$ )

Mentre su BDD cambiare l'ordine  
può semplificarmi o complicare il  
processo

## Dispositivo di tipo Multiplexer (MPX)

giovedì 22 ottobre 2020 17:37

Può essere visto come un **SCINTILLATORE**: in base al segnale, emette un impulso elettrico che si muove da una sorgente rispetto ad un'altra (diverse uscite)



porta in uscita un valore in base ai valori di ingresso e commuto (distribuisce, dirige) il segnale in base al valore di controllo.

c → voglio uscire come controllo

$c \begin{cases} 0 \\ 1 \end{cases}$  ) due comportamenti diversi  $\rightarrow c = 0 \rightarrow u = b$

$$c = 1 \rightarrow u = a$$

Posso pensarlo come un testo  
punto sì/no

Come realizzarlo?

ES 1

- Posso definire il comportamento su una TAV DI VERITÀ

c → b		u
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	0
1	1	0
1	1	1



e' una rappresentazione ridondante, se scindiamo

più valori per rappresentare gli stessi in uscita

a b / c		0	1
a	b	0	1
0	0	0	0
0	1	1	0
1	1	1	1
1	0	0	1

RAPPRESENTAZIONE ALGEBRICA:

$$b \cdot \bar{c} + a \cdot c$$

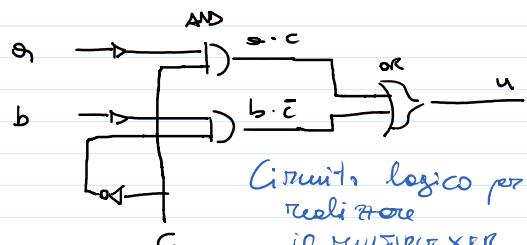
indipendente da a

indipendente da b

N.B.

Il circuito logico qui rappresentato (in forma grafica) è il **massimo livello di dettaglio** a cui mi fermo. Se volessi proseguire, e quindi realizzare fisicamente il dispositivo, concretizzando i nodi AND, OR ecc..., dovrei trattare l'hardware. Dunque TRANSISTOR, IMPULSI ELETTRICI, dissipazione del calore generato, occupazione di spazio, consumi di ENERGIA.

RAPPRESENTAZIONE GRAFICA MINIMALE

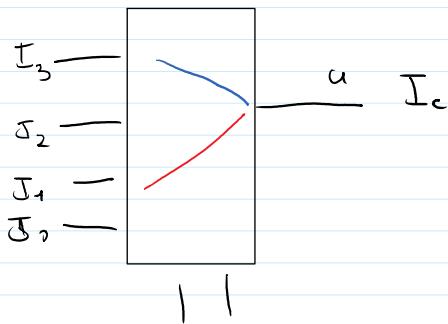


Circuito logico per realizzazione del MULTIPLEXER

Lo ha preso si rivolge alle "evidenze" studiate in ELETTRONICA.

ES 2

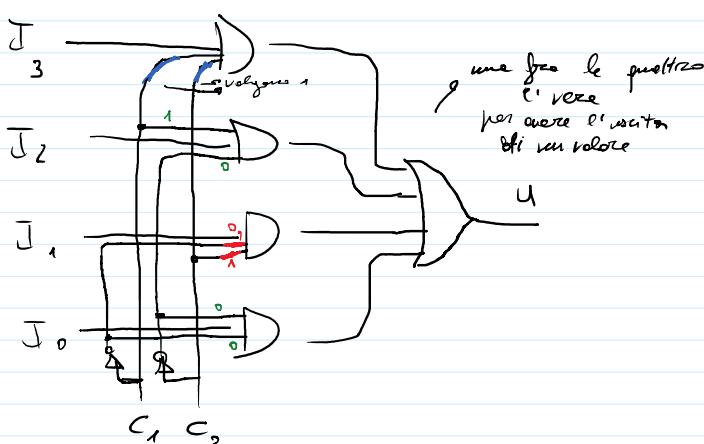
Se vogliamo realizzare un multiplexer con più di due ingressi?



$C_1 \ C_0 \rightarrow$  non puoi connettere multidiceto su un solo filo, c' deve essere più BIT  
 (tanti quanti i valori che ammettono gli ingressi)  
 $\begin{matrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{matrix} = 4$   
 $\hookrightarrow 2 \text{ BIT}$  (coppia 4 combinazioni, a cui posso associare gli ingressi)  
 e associo al valore 1 il collegamento  $J_1 - I_c$   
 al valore 3 " "  $J_3 - I_c$

$$\text{n° BIT} = \log_2 (\text{n° Ingressi})$$

Realizzazione:



F

N.B.

- dispositivo più complesso del precedente;

- = più consumo
- = più spazio occupato
- = più colore
- = più componenti

perché:

- ha 2 NOT, 4 ingressi
- un OR più grande

FS 3

8 AND

Per costruire con 8 ingressi posso duplicare questo

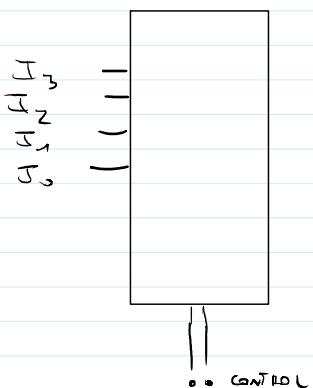
schema, aggiungendo un nuovo ingresso di

controllo  $C_2$  che assume 1 per 4 ingressi 0 per gli

altri 4 ingressi (TOTALE 3 BIT  $\rightarrow 2^3$  COMBINAZIONI, una per ogni ingresso).

## Dispositivo Demultiplexer (DPX)

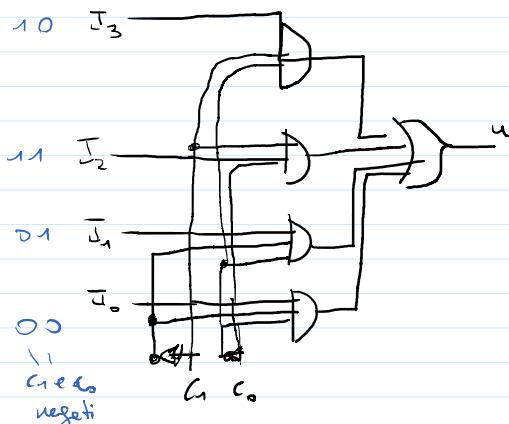
venerdì 23 ottobre 2020 16:36



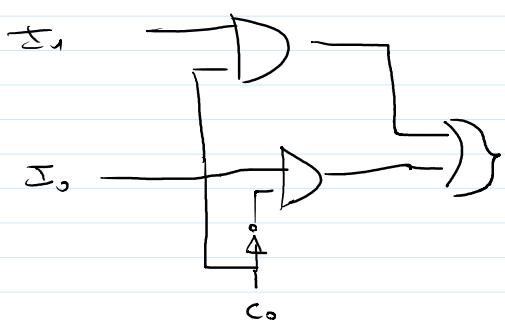
Dispositivo tipo MUX

[Ciprolog]

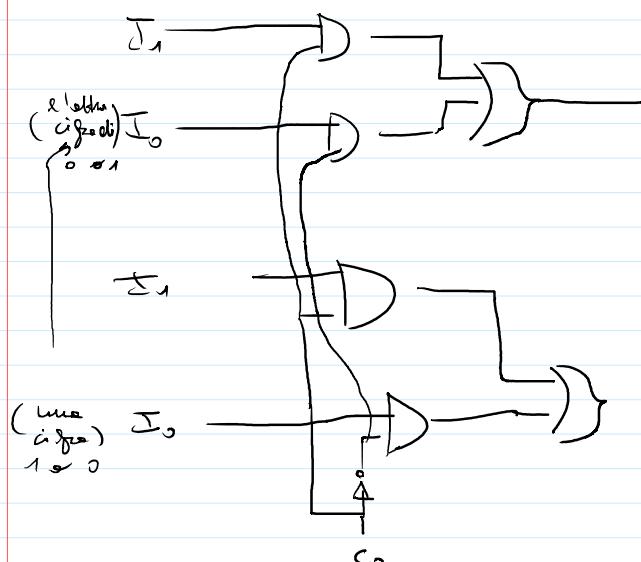
per ogni ingresso = 1 funzione AND



Estensione (più ingressi)



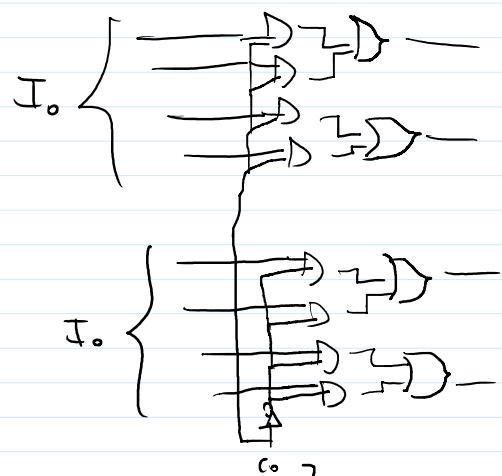
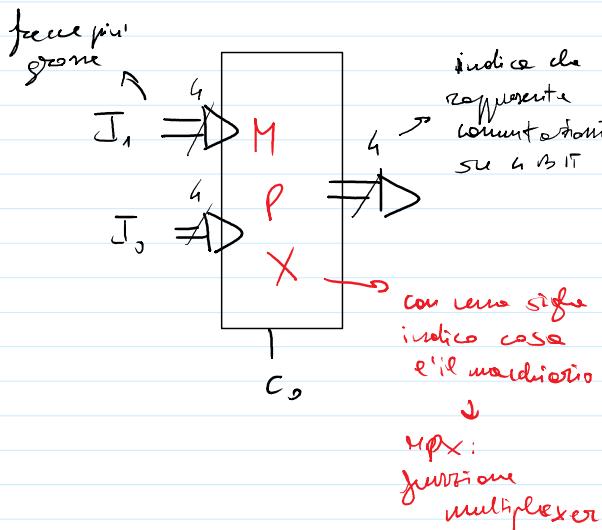
Con 3 BIT nel controllo, replica la struttura sopra  
altre due volte:



Dentro una scatola:

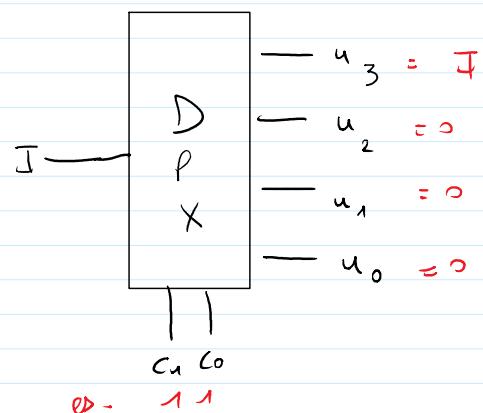
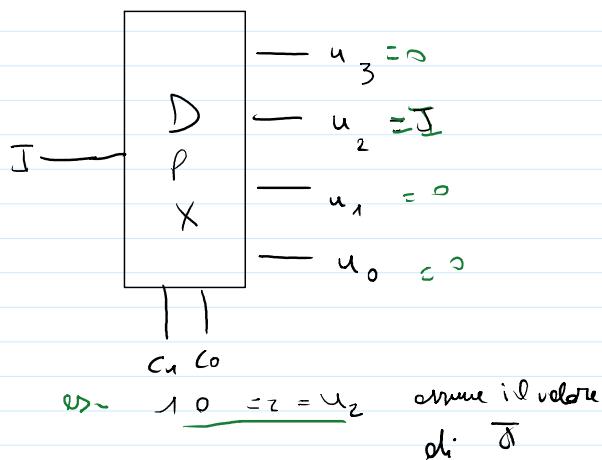


de corrisponde a

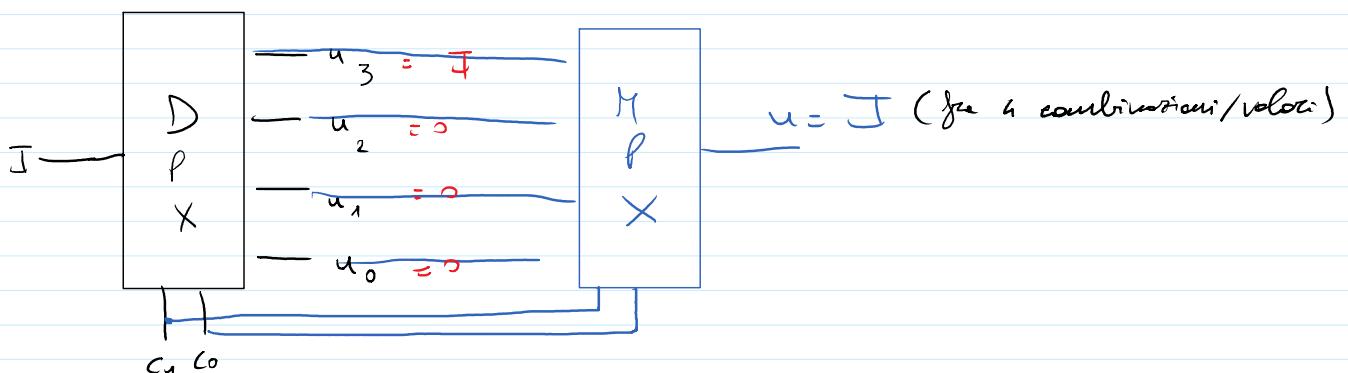


quelle più significative (3 bit) decidono che ingresso utilizzare

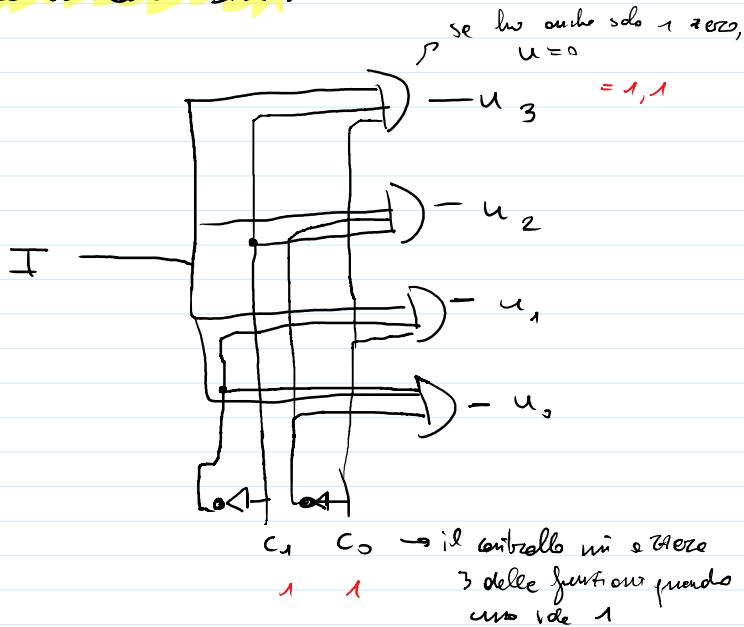
### FUNZIONE INVERSA : Demultiplexer



E' della funzione inversa: con MPX posso raccolgere uno fra n valori, mentre DMUX summa un ingresso fra n valori



### Realizzazione DFX:



e differenze di MPX

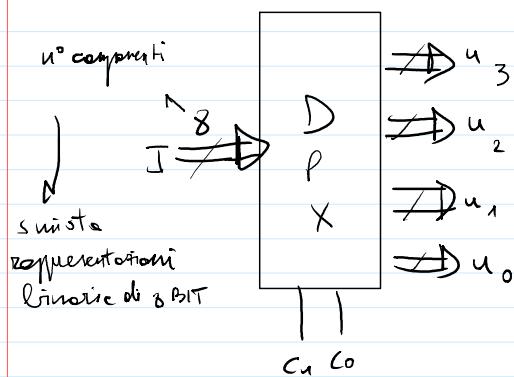
la parte centrale è

regolare, ma le uscite  
dell'AND nel MPX

hanno una funzione OR.

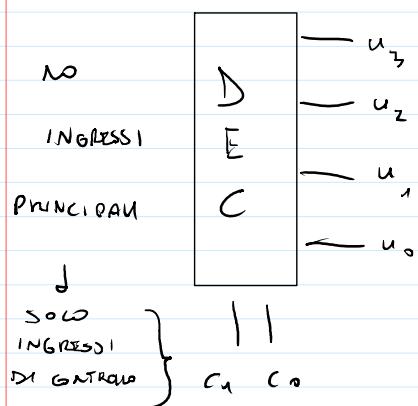
Inoltre ho un n° di  
ingressi differente.

N.B. anche qui il DFX puo' lavorare con le estensioni :



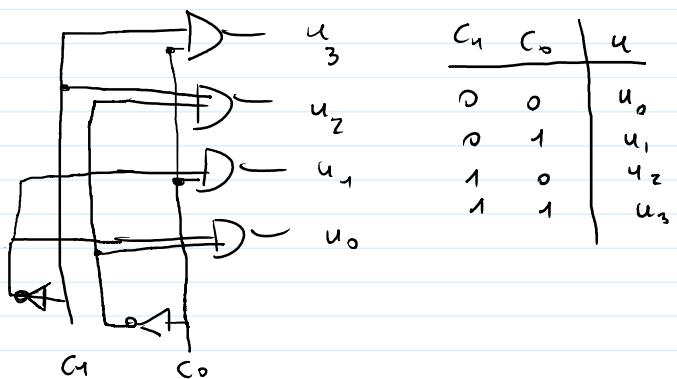
## Dispositivo Decoder (DEC)

venerdì 23 ottobre 2020 17:11



Quando un' uscita è 1, le altre sono zero.

Circuito logico uguale al DPX (meno per gli ingressi)



Il dec è una semplificazione del DPX

nel caso I fosse fisso

Anche in questo caso posso espandere il DEC.