Capitolo 3
Reti Combinatorie

Reti Logiche T

3.1 Combinatorio vs. Sequenziale

La rete logica

i ∈ I: alfabeto di ingresso

u ∈ U: alfabeto di uscita

F: relazione di ingresso/uscita o di causa/effetto

Rete logica -Modello matematico che assume come *primitive* alcune semplici modalità di elaborazione di segnali binari e deduce da queste in modo rigoroso

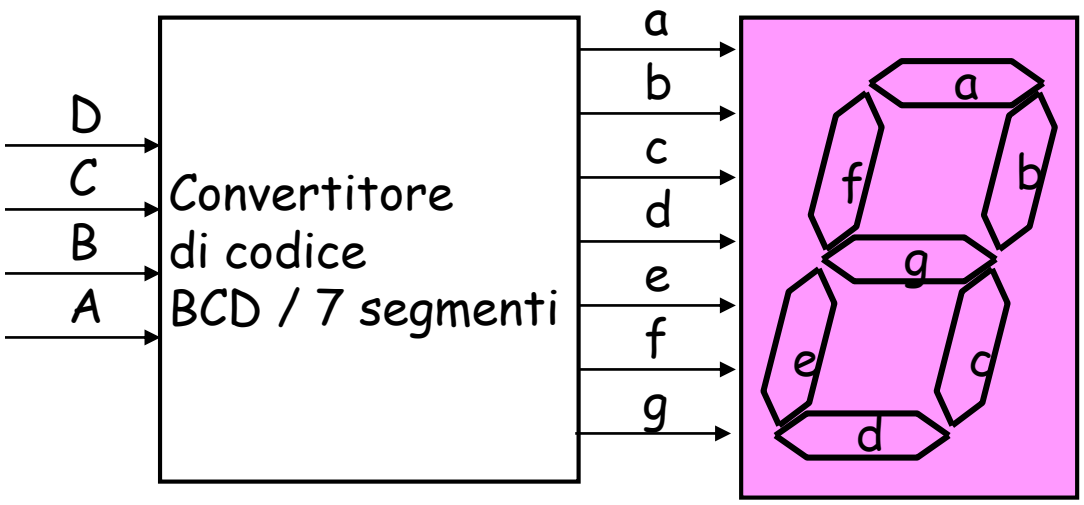
- quale struttura soddisfa un dato comportamento (*sintesi*)
- quale comportamento ha una data struttura. (*analisi*)

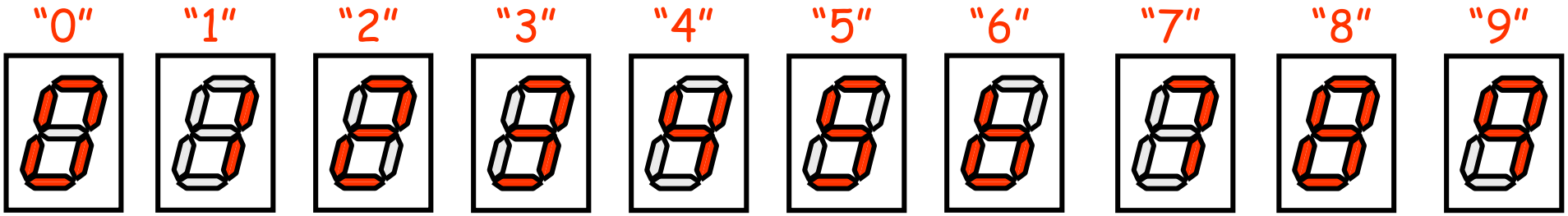
Combinatoria vs. Sequenziale

- **Rete (o macchina) combinatoria**: l'uscita dipende unicamente dagli ingressi correnti

- **Rete (o macchina) sequenziale**: l'uscita non è univocamente determinata dagli ingressi correnti, ma dipende anche dalla storia passata (sequenza di ingressi visti in precedenza) e dallo scorrere del tempo

Il convertitore BCD/7 Segmenti





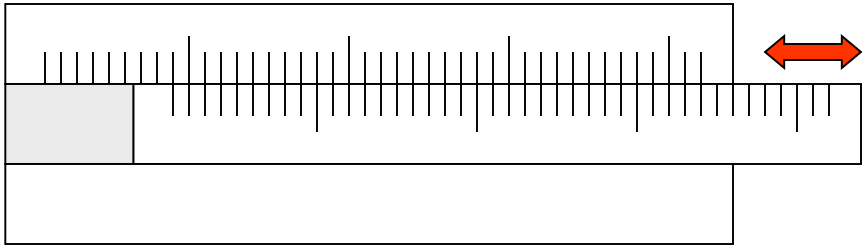
- Questo convertitore è un esempio di macchina combinatoria: le 7 uscite (abcdefg) sono univocamente determinate dal valore corrente dei 4 ingressi (ABCD)
- Es: ABCD=0001 -> abcdefg = 1001111

Altri esempi di macchine combinatorie

×	1	2	3	….	8	9
1	1	2	3	….	8	9
2	2	4	6	….	16	18
3	3	6	9	….	24	27
……						
……						
8	8	16	24	….	64	72
9	9	18	27	….	72	81

la tavola pitagorica

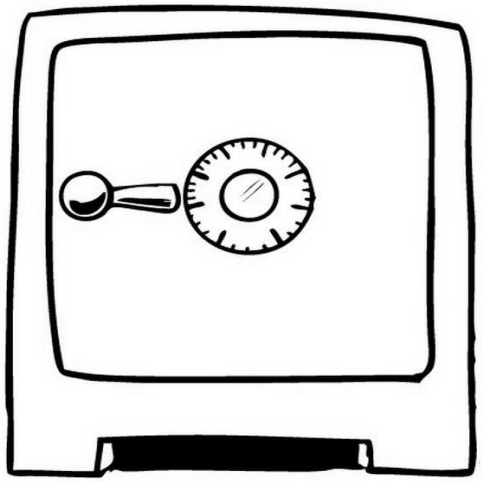
il dizionario



il regolo

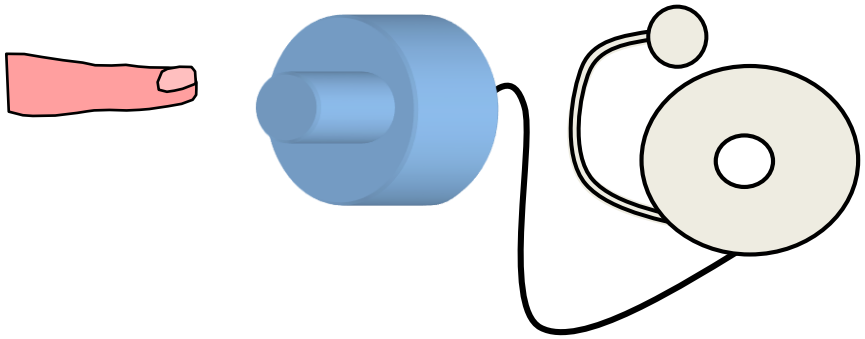
La cassaforte

- Macchina con 2 ingressi (x_0,x_1) e una uscita (**z**)
- z=0/1: cassaforte chiusa/aperta
- z=1 con ingresso 11 se e solo se la sequenza in ingresso è quella corretta (x_0x_1 = 00 – 10 – 11)
- (è un esempio di *riconoscitore di sequenze*)



- Con ingresso 11, z = 0 o 1 ??
  L'informazione è insufficiente a determinare il comportamento della macchina, **dipende** anche dalla storia

Il campanello



i: Pulsante	*u: Suoneria*
Premuto	din
Rilasciato	nessun suono

$u = F(i)$

Macchina combinatoria

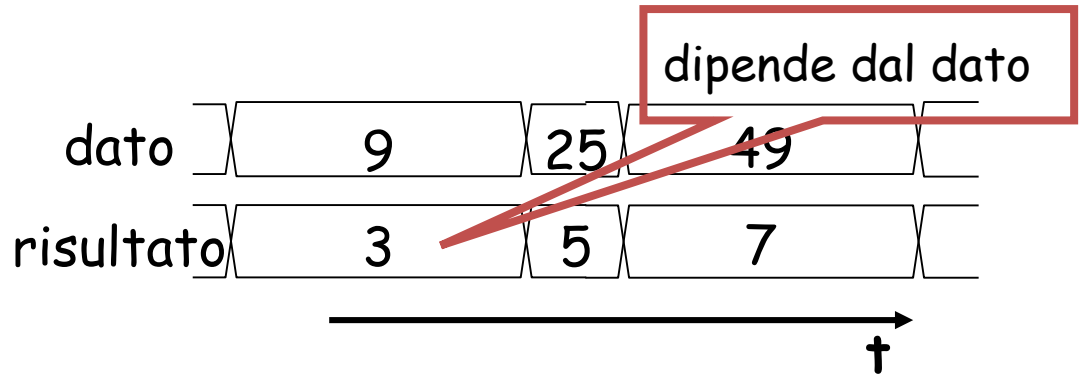
	i: Pulsante	*u: Suoneria*
t_0	Premuto	din
t_1	Rilasciato	nessun suono
t_2	Rilasciato	don
t_3	Rilasciato	nessun suono

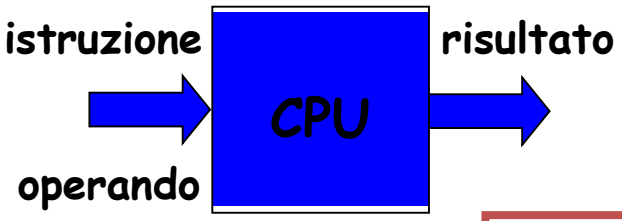
$u(t_i) = F(i(t_i), i(t_{i-1}), ..)$

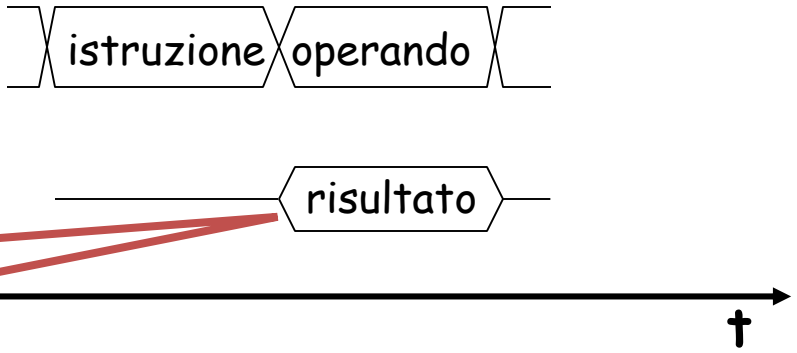
Macchina sequenziale: a parità d'ingresso, risposte diverse ad istanti diversi

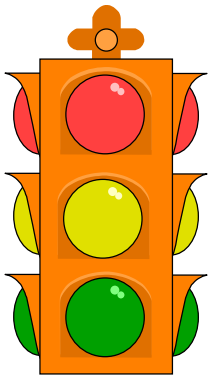
Altri Esempi

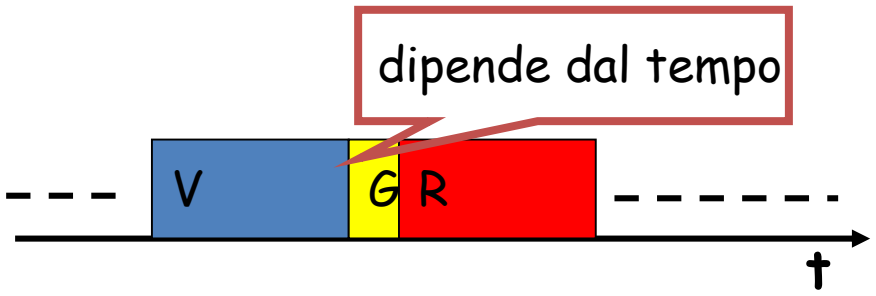
$u = \sqrt{i}$





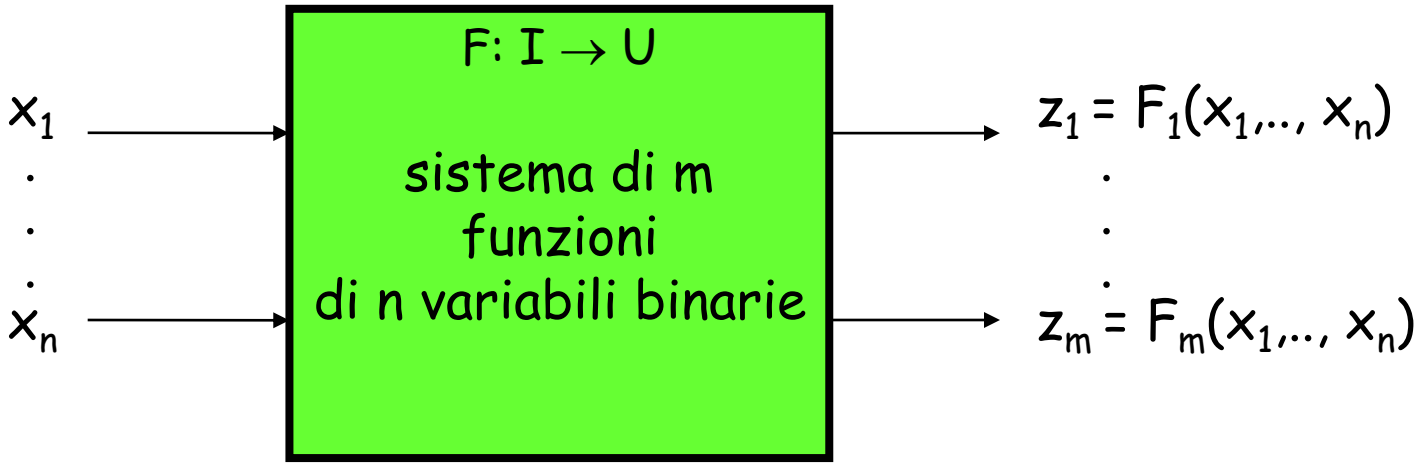






3.2 Funzioni, espressioni e schemi logici

Rete logica combinatoria

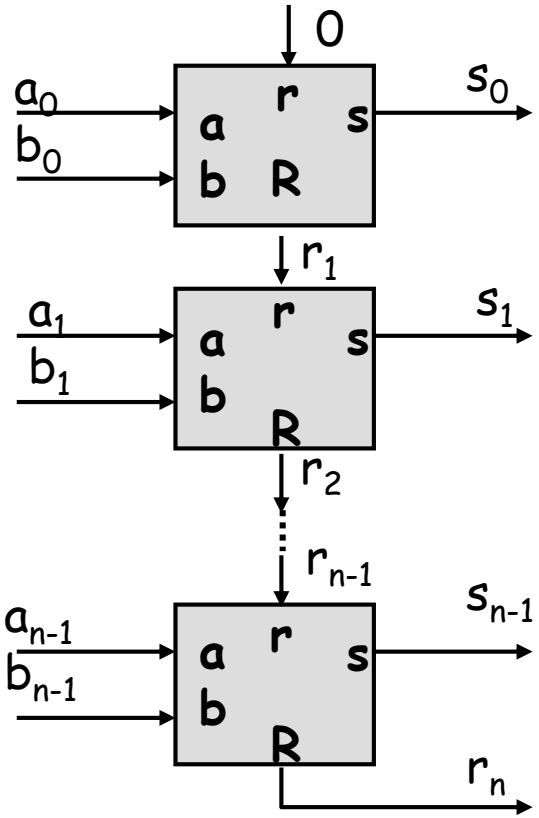


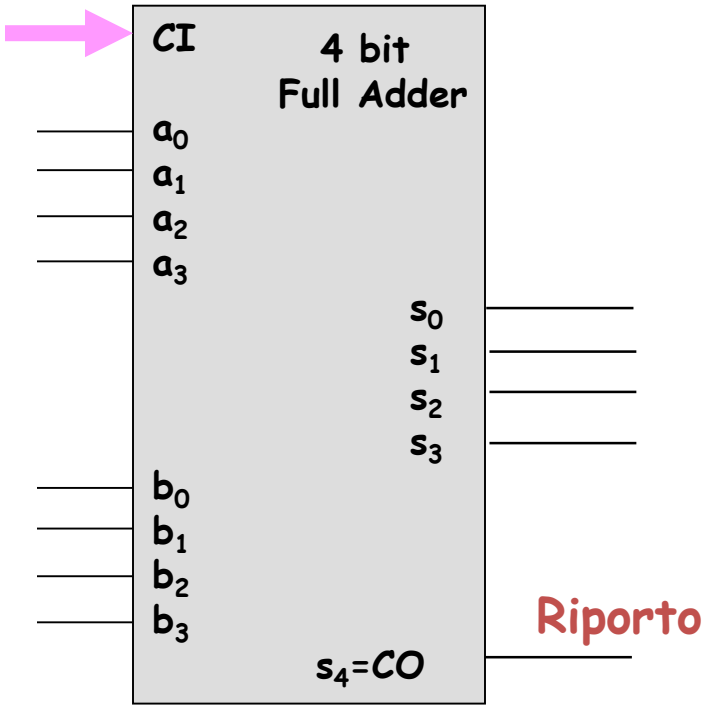
Rete logica combinatoria
i valori delle variabili di uscita (variabili dipendenti)
dipendono solo dai valori contemporanei
delle variabili di ingresso (variabili indipendenti)

Composizione e decomposizione

La disposizione **in serie e/o in parallelo** di macchine combinatorie è ancora una macchina combinatoria

- Questa proprietà era stata sfruttata in precedenza per realizzare un addizionatore a n bit disponendo in serie FA a 1 bit



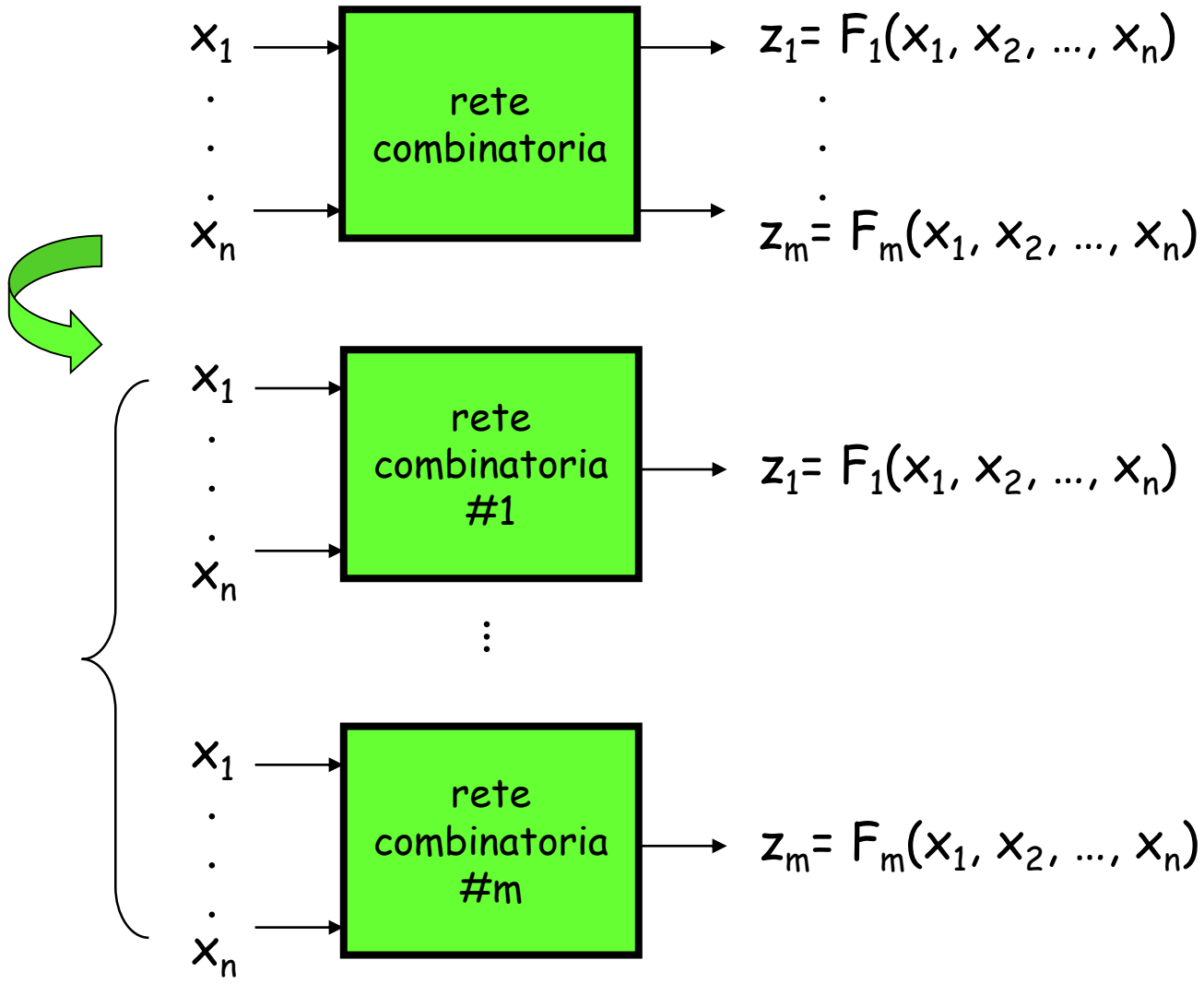


N.B. Se non si considera il riporto, l'addizione è fatta modulo 16

CI = 0: $S_2 = (A + B)_2$
CI = 1: $S_2 = (A + B + 1)_2$

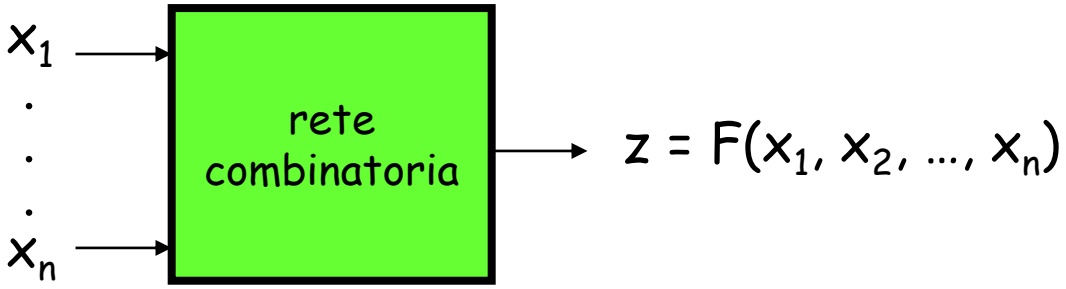
Decomposizione di una rete combinatoria

- Sfruttando la proprietà precedente, una rete combinatoria con m uscite può essere decomposta in m reti con un'unica uscita



Comportamento & Struttura

- Nel caso di reti combinatorie, è possibile passare da descrizione del comportamento a realizzazione (e viceversa) mediante analisi e sintesi



Comportamento

Struttura

$z = F(x_1, \ldots, x_n)$

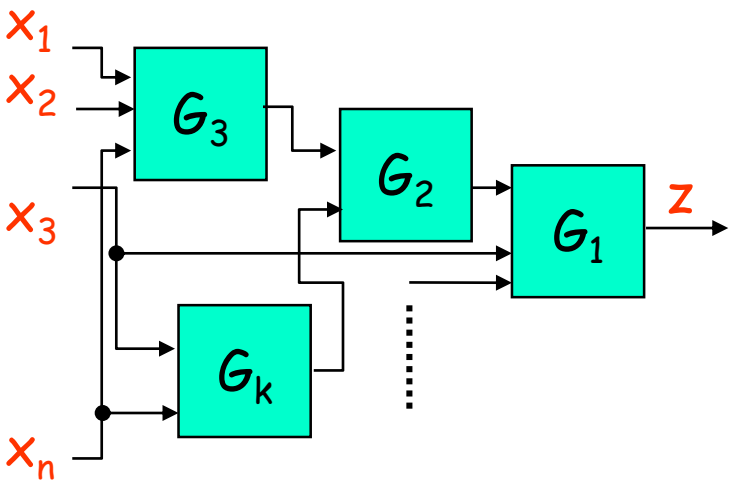
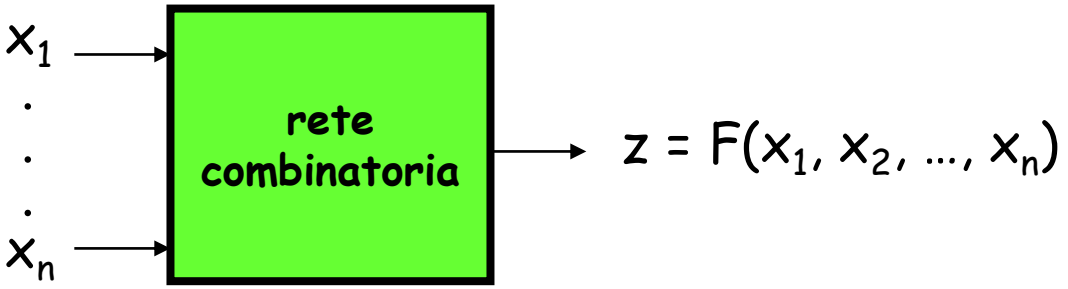


Tabella della verità

- Descrive univocamente il comportamento di una rete combinatoria a n ingressi (ovvero di una funzione F di n variabili binarie $x_1, x_2, \ldots, x_n$)



n+1 colonne

2^n righe

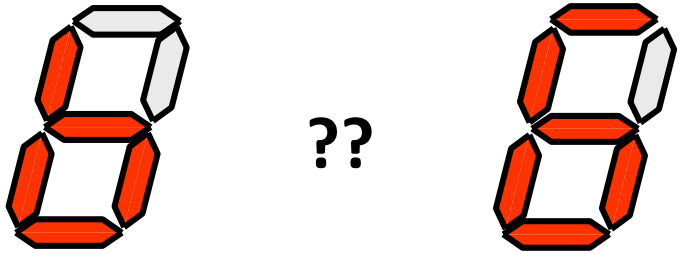
$x_1\ x_2\ \ldots\ x_n$	$F(x_1, x_2, \ldots, x_n)$
0 0 …… 0	0 *oppure* 1 *oppure* -
0 0 …… 1	0 *oppure* 1 *oppure* -
⋮	⋮
1 1 …… 0	0 *oppure* 1 *oppure* -
1 1 …… 1	0 *oppure* 1 *oppure* -

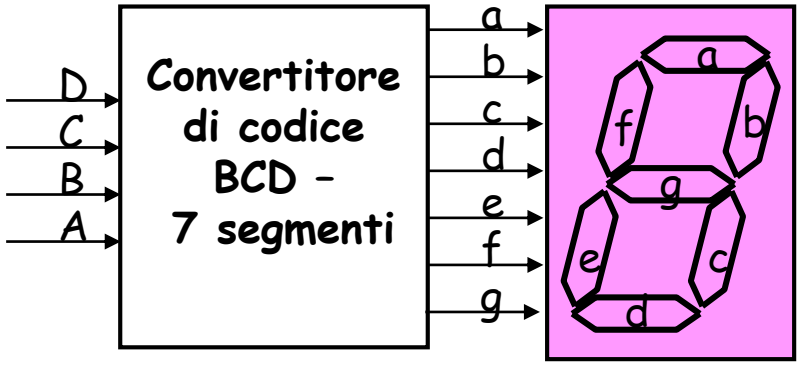
Funzione completa

Funzione incompleta

Il convertitore BCD/7 Segmenti - reloaded

- Descrizone a parole non è univoca: «6» a cosa equivale?





La tabella della verità ci permette di descrivere inequivocabilmente il comportamento di una macchina combinatoria

Funzioni complete e incomplete

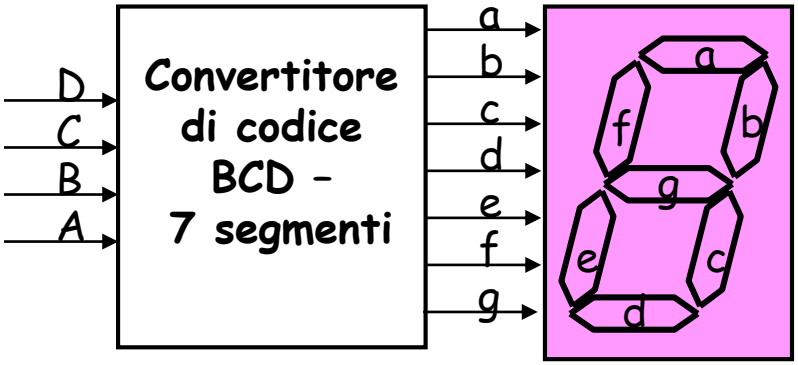
Funzione completa di n variabili binarie $z = F(x_1, x_2, \ldots, x_n)$:
insieme di 2^n coppie ordinate $\{x, z \mid x \in B^n, z \in B\}$ formate da una configurazione di valori delle variabili indipendenti x_i e dal corrispondente valore della variabile dipendente z.

Funzione incompleta o non completamente specificata:
il dominio è un sottoinsieme di B^n
(vi sono configurazioni di ingresso non specificate)

Esempio: convertitore di codice BCD $\rightarrow$ 7 segmenti
(v. prossima slide)

Convertitore BCD/7 Segmenti – parte III

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0



..e le rimanenti configurazioni di ingressi?

1	0	1	0	-	-	-	-	-	-	-
1	0	1	1	-	-	-	-	-	-	-
1	1	0	0	-	-	-	-	-	-	-
1	1	0	1	-	-	-	-	-	-	-
1	1	1	0	-	-	-	-	-	-	-
1	1	1	1	-	-	-	-	-	-	-

utilizzo per l'uscita un ulteriore simbolo che rappresenta una **condizione di «indifferenza»** (*don't care*)

Funzioni complete di una variabile

x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

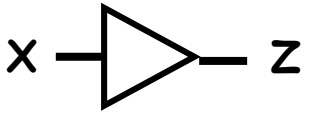
4 funzioni di una variabile

f_0, f_3 : costanti 0 e 1
f_1: identità o buffer
f_2: not

X	Z
0	0
1	1

BUFFER

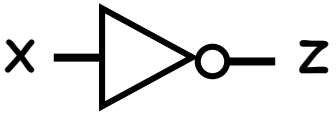
Z = X



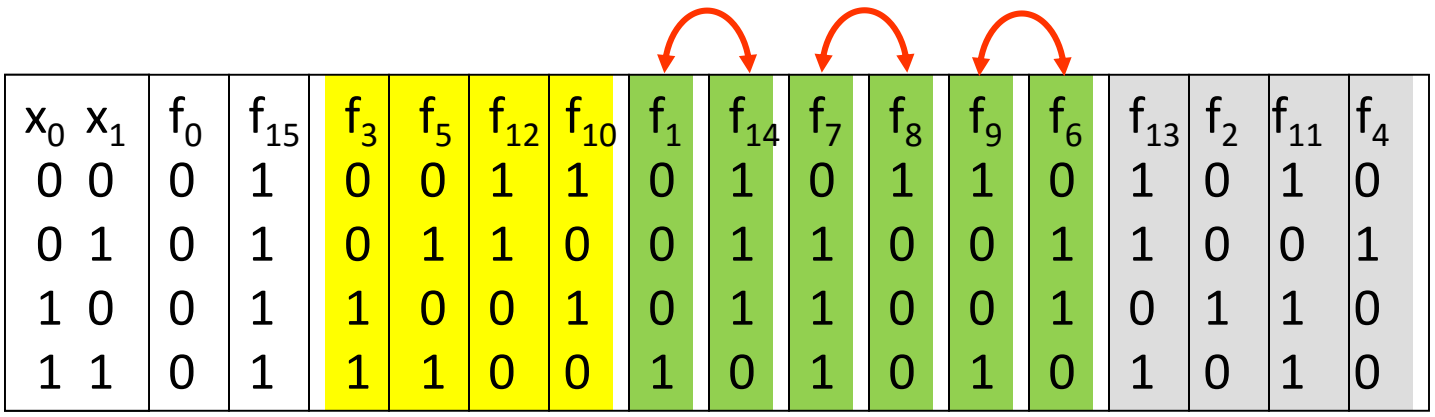
X	Z
0	1
1	0

NOT

Z = X'



Funzioni complete di due variabili



x_0	x_1	f_0	f_{15}	f_3	f_5	f_{12}	f_{10}	f_1	f_{14}	f_7	f_8	f_9	f_6	f_{13}	f_2	f_{11}	f_4
0	0	0	1	0	0	1	1	0	1	0	1	1	0	1	0	1	0
0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1	0	1	1	0	0	1	0	1	1	0
1	1	0	1	1	1	0	0	1	0	1	0	1	0	1	0	1	0

16 funzioni di due variabili

f_0, f_{15} : costanti 0 e 1
f_3, f_5 : identità o buffer
f_{12}, f_{10} : not

f_1 : and
f_{14} : nand
f_7 : or
f_8 : nor
f_9: equivalence
f_6: ex-or

f_{13}: x_0=1 implica x_1=1
f_{11}: x_1=1 implica x_0=1
f_2: complemento di f_{13}
f_4 : complemento di f_{11}

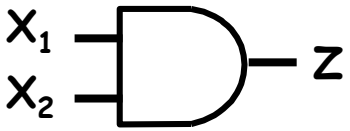
funzioni complementari

Funzioni elementari & Porte logiche ...

X_1	X_2	Z
0	0	0
0	1	0
1	0	0
1	1	1

AND

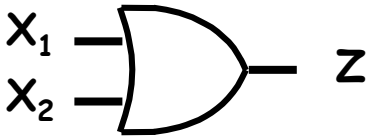
$Z = X_1 \cdot X_2$



X_1	X_2	Z
0	0	0
0	1	1
1	0	1
1	1	1

OR

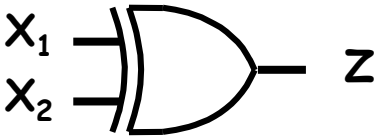
$Z = X_1 + X_2$



X_1	X_2	Z
0	0	0
0	1	1
1	0	1
1	1	0

EXOR

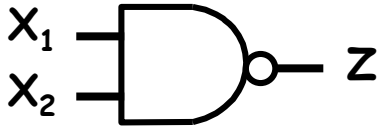
$Z = X_1 \oplus X_2$



... Funzioni elementari & Porte logiche

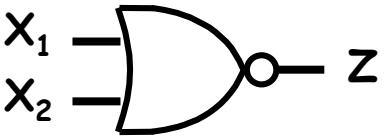
X_1	X_2	Z
0	0	1
0	1	1
1	0	1
1	1	0

NAND $\quad Z = X_1 \uparrow X_2$



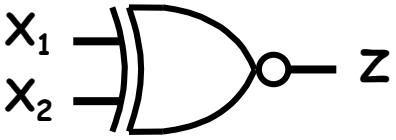
X_1	X_2	Z
0	0	1
0	1	0
1	0	0
1	1	0

NOR $\quad Z = X_1 \downarrow X_2$



X_1	X_2	Z
0	0	1
0	1	0
1	0	0
1	1	1

EQUIVALENCE $\quad Z = (X_1 \equiv X_2)$



Implicazione logica

- Date due proposizioni A e B, «A implica B» significa che se A è vera, **allora** è vera anche B
  - «sta piovendo» implica «è nuvoloso»

- NOTA: per l'implicazione logica, «A falsa» implica sia «B vera» che «B falsa» (se l'ipotesi è falsa, la tesi è sempre valida!)
  - «Napoleone ha vinto a Waterloo» implica «Domani sarà una bella giornata»

X_1	X_2	Z
0	0	1
0	1	1
1	0	0
1	1	1

X1 implica X2

$Z = X_1 \Longrightarrow X_2$

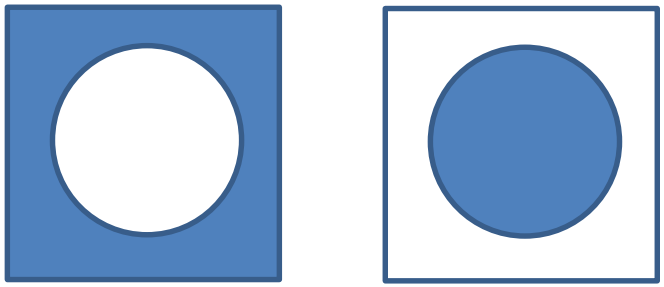
X_1	X_2	Z
0	0	1
0	1	0
1	0	1
1	1	1

X2 implica X1

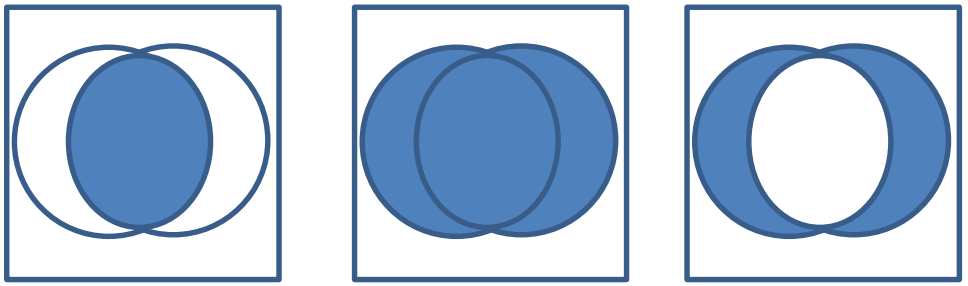
$Z = X_2 \Longrightarrow X_1$

Diagrammi di Venn

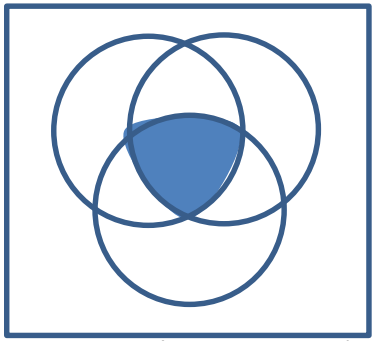
- Rappresentazione alternativa di funzioni booleane di n variabili
- Il rettangolo esterno rappresenta il dominio della funzione
- Ogni variabile è rappresentata da un cerchio (all'interno della quale tale variabile vale 1)
- Le aree «colorate» indicano le aree del dominio per cui la funzione assume valore 1.



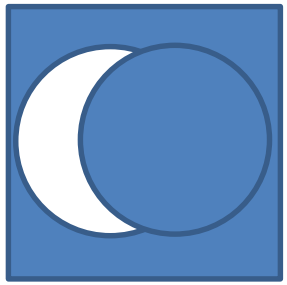
Funzioni di 1 variabile (not, buffer)



Funzioni di 2 variabili (and, or, ex-or)



Funzioni di 3 variabili (and a 3 ingressi)



X1 implica X2 (diagramma di Venn)

Funzioni complete di n variabili

Il numero di distinte funzioni di n variabili binarie

$$\Phi(n) = 2^{2^n}$$

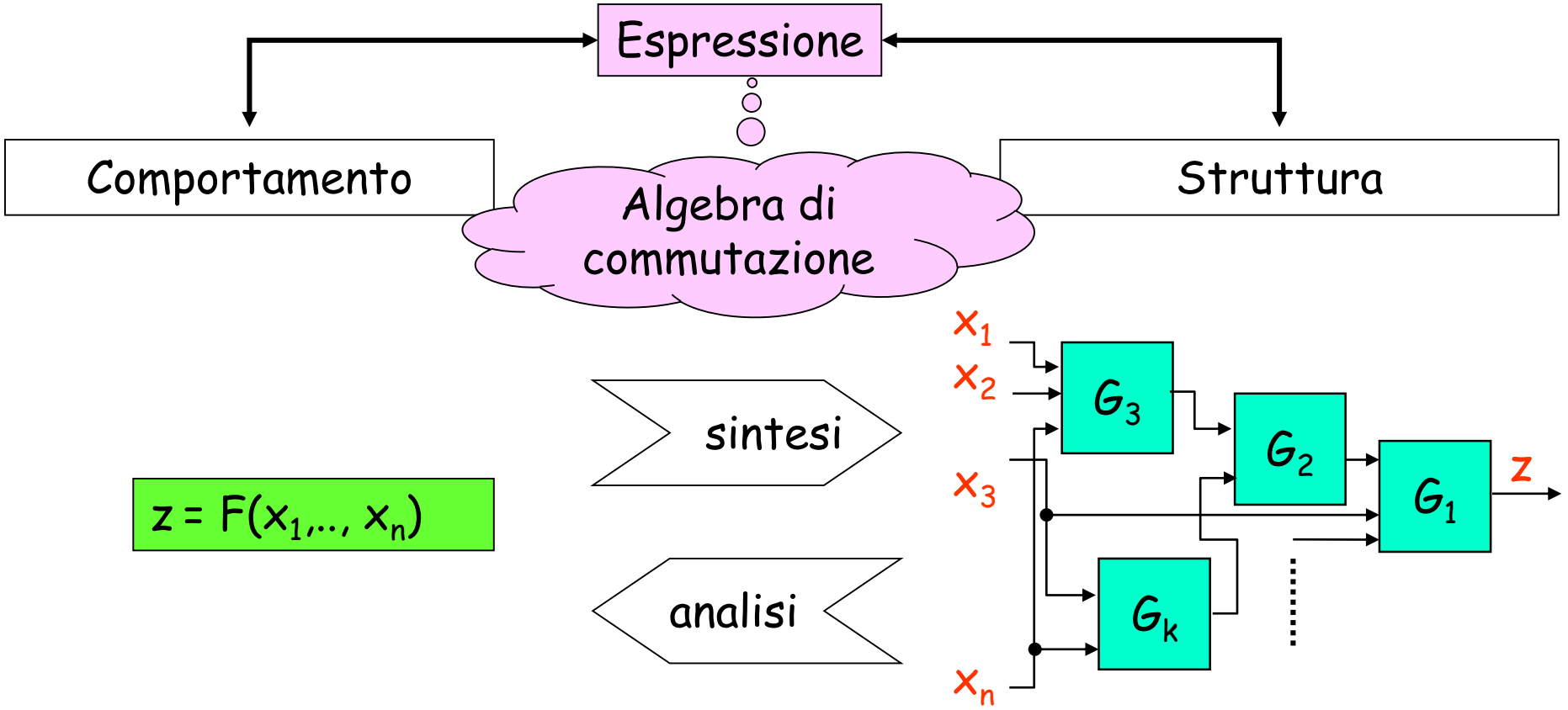
aumenta esponenzialmente con n

n	Φ(n)
1	4
2	16
3	256
4	65536
…	…

Non realizzazione ad hoc di ogni funzione, bensì **opportuna composizione di componenti disponibili in forma integrata**

SSI | MSI | LSI,VLSI

Comportamento ↔ Struttura



- L'algebra di commutazione è il «*ponte*» che ci permette di passare dal dalla descrizione del comportamento di una rete combinatoria alla sua struttura e viceversa
- Ad esempio, durante il processo di sintesi è possibile manipolare e semplificare le espressioni a livello algebrico per ottenere una realizzazione circuitale più efficiente, o meno costosa

Algebre binarie

Algebra binaria - Sistema matematico formato da un insieme di operatori definiti assiomaticamente ed atti a descrivere con una espressione ogni funzione di variabili binarie

Calcolo delle proposizioni
{*vero*, *falso*} {*e*, *o*, *non*}
tre operatori

Crisippo (250 a.c.)
G. Boole (1854)

AND, OR e NOT costituiscono un *insieme funzionalmente completo*

Algebra di commutazione
{0, 1} {+, . , '}
tre operatori

C. Shannon (1938)

Algebra del nand
{0, 1} {↑}
un operatore

Algebra del nor
{0, 1} {↓}
un operatore

Algebra lineare
{0, 1} { ⊕ , .}
due operatori
(EX-OR, AND)

Calcolo delle proposizioni

- Assegnata una qualsiasi descrizione a parole di una funzione di n variabili binarie, tramutarla in una espressione contenente solo le operazioni logiche eseguite da certo insieme di gate (es. AND, OR e NOT)

Proposizione - Frase o "vera" o "falsa", formata da affermazioni o "vere" o "false" unite dai connettivi o, e, non.

"vero" → 1, "falso" → 0, "e" → ., "o" → +, "non" → '

- Es: la frase «F(x,y) vale 1 se o x vale 1 o y vale 1»
  - descrive la funzione "or"
  - è equivalente alla proposizione "o x o y" (*vera* per 01,10,11 e *falsa* per 00)
  - è equivalente all'espressione x + y

- Es. : la frase «F(A,I_0,I_1) vale 1 se o (non A e I_0) o (A e I_1)» (*selettore a 2 vie*):
  - è equivalente all'espressione $A' \cdot I_0 + A \cdot I_1$

A	I_0	I_1	U
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Calcolo delle proposizioni: esercizio

La selezione dell'equipaggio per una missione spaziale, indicati con A, B, C, D, E i potenziali candidati, deve essere operata in modo tale da soddisfare tutte le seguenti condizioni:

- l'equipaggio deve comprendere o A o B, ma non entrambi;
- l'equipaggio deve comprendere o C o E o entrambi;
- l'equipaggio deve comprendere sia A che C, o nessuno dei due;
- se D fa parte dell'equipaggio, anche B deve farne parte;
- se E fa parte dell'equipaggio, anche C e D devono farne parte.

Una certa polizza assicurativa P può essere emessa solo se il richiedente soddisfa almeno una delle seguenti condizioni:

- è un uomo che non ha ancora 25 anni;
- è sposato ed ha 25 anni o più;
- è un uomo sposato che già possiede la polizza Q;
- è una donna sposata che non possiede la polizza Q;
- è sposato, non ha ancora 25 anni e possiede già la polizza Q.

3.3
Algebra di commutazione

Algebra di commutazione

Sistema matematico

1) un insieme di simboli $B \equiv \{0, 1\}$

2) un insieme di operazioni $O \equiv \{+, ., '\}$

somma logica (+) prodotto logico (.) complementazione (')

3) un insieme di postulati P:

$0 + 0 = 0$	$0 . 0 = 0$	$0' = 1$
$1 + 0 = 1$	$1 . 0 = 0$	$1' = 0$
$0 + 1 = 1$	$0 . 1 = 0$	
$1 + 1 = 1$	$1 . 1 = 1$	

Costanti, Variabili, Espressioni

Costanti: elementi 0, 1 dell'insieme B
Variabili: entità suscettibili di assumere il valore 0 o 1
Espressioni: stringhe finite di costanti, variabili, operatori e parentesi, formate in accordo alle seguenti regole:
1) 0 e 1 sono espressioni
2) una variabile è una espressione
3) se A è un'espressione, lo è anche (A')
4) se A, B sono espressioni, lo sono anche (A+B), (A.B)

Esempi: a+(b.c) a + bc

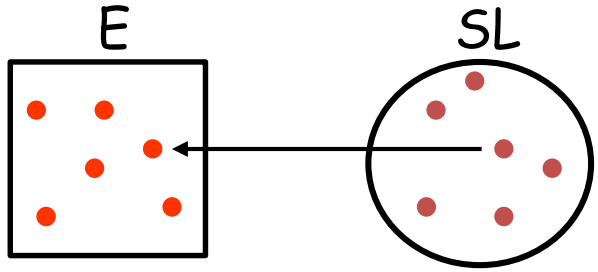
a'.b (a+b)' a'b + 0 + ab'

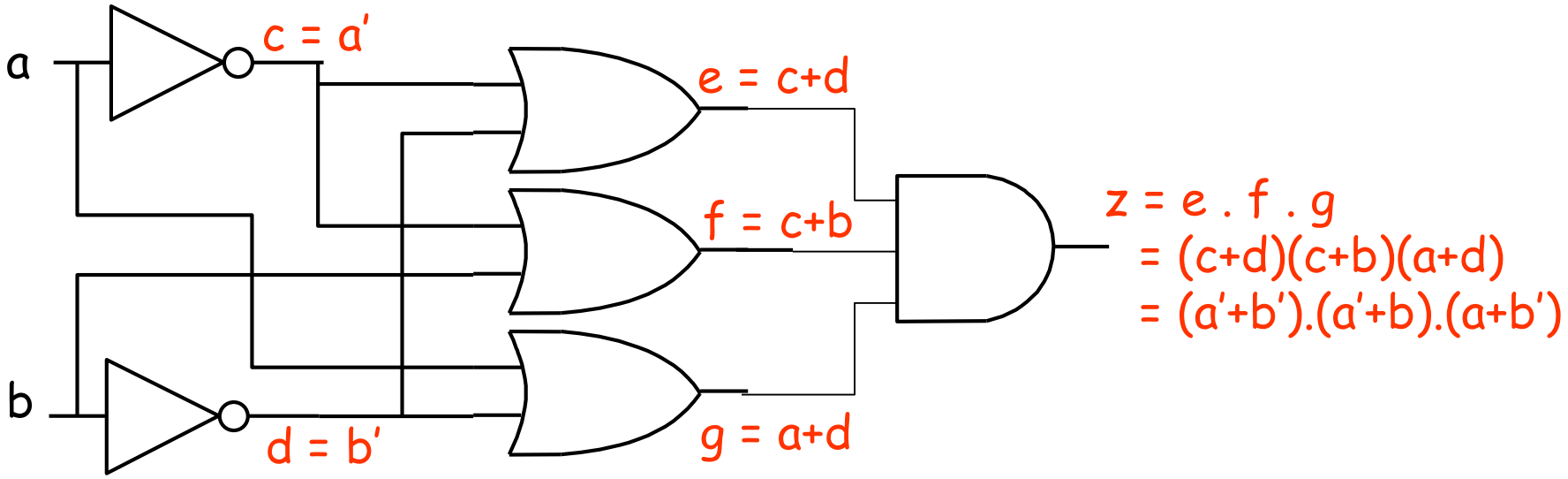
N.B. - L'operazione di prodotto è prioritaria rispetto alla somma e non è obbligatorio racchiuderla tra parentesi. La notazione AB indica A·B

Schemi logici e Espressioni

Schema logico - Descrizione grafica di una struttura formata da simboli di gate e da collegamenti tra le loro linee di ingresso e di uscita.

I) Ogni struttura formata da gate connessi in serie e/o in parallelo è descritta da una sola espressione.

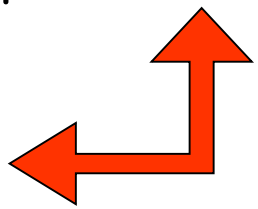




Valutazione di una espressione

Valutazione di una espressione di n variabili per una n-pla di valori
1) Si sostituisce ad ogni variabile il valore che le compete.
2) Partendo dalle parentesi più interne, si sostituisce ogni operazione con il suo risultato fino ad ottenere o la costante 0 o la costante 1.

Esempio: E(a,b,c) = a+(b.c) per a=0, b=1, c=0
= 0+(1.0)
= 0+0
= 0



N° di valutazioni - Una espressione di n variabili può essere valutata in 2^n modi diversi.

- Quindi, valutando una espressione di n variabili per tutte le possibili 2^n configurazioni, ottengo la descrizione della funzione ad essa associata

Espressioni e Funzioni

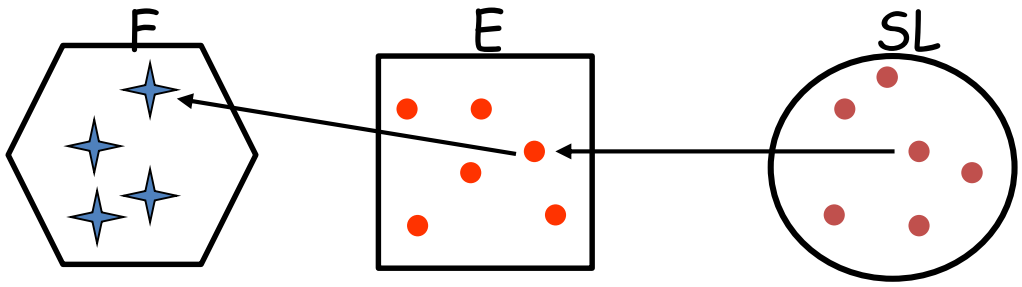
II) Ogni espressione descrive una e una sola funzione completa.

Le 2^n valutazioni di una espressione $E(x_1, x_2, \ldots, x_n)$ creano 2^n coppie x, z $\{x, z \mid x \in B^n, z \in B\}$

Esempio: E(a,b,c) = a+(b.c)

E(0,0,0) = 0+(0.0) = 0
E(0,0,1) = 0+(0.1) = 0
E(0,1,0) = 0+(1.0) = 0
E(0,1,1) = 0+(1.1) = 1
E(1,0,0) = 1+(0.0) = 1
E(1,0,1) = 1+(0.1) = 1
E(1,1,0) = 1+(1.0) = 1
E(1,1,1) = 1+(1.1) = 1

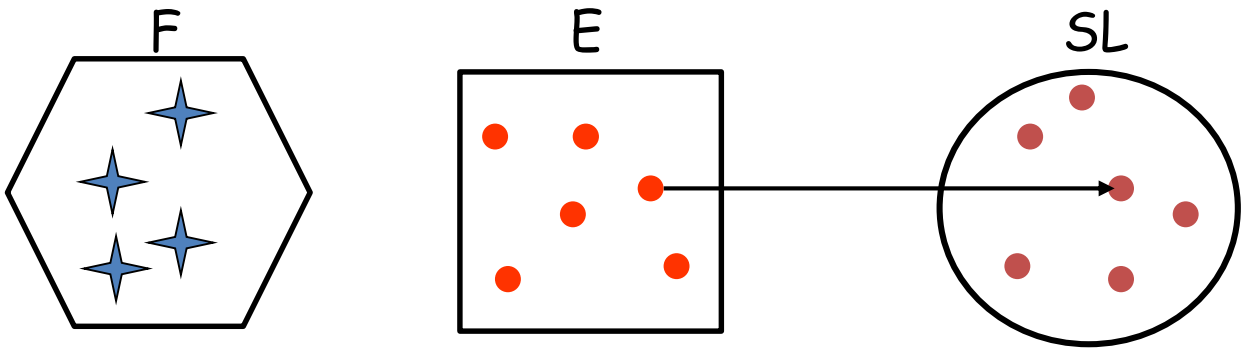
a b c	E
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	1



ANALISI: partendo da un preciso schema logico, si può determinare direttamente la funzione ad esso corrispondente

Espressioni e Schemi logici

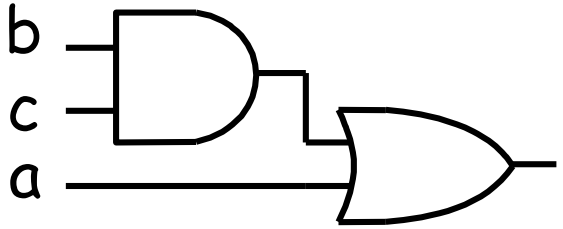
III) Ogni espressione descrive una sola struttura formata da gate connessi in serie e/o in parallelo.



Per individuare lo schema descritto da una espressione:
1) si parte dalle parentesi più interne e si traccia il simbolo del gate corrispondente all'operazione, collegandone gli ingressi ai segnali esterni;
2) si procede in modo analogo con le altre coppie di parentesi, considerando via via come ingressi dei nuovi gate anche le uscite di quelli già tracciati.

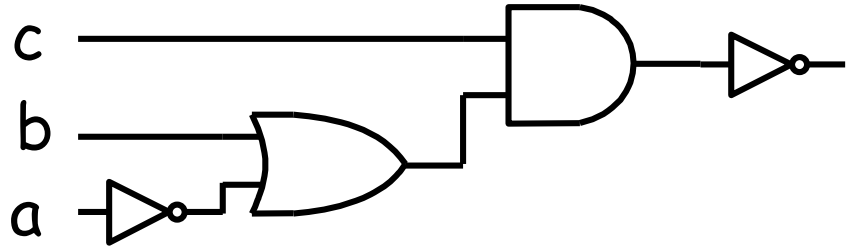
Esempio:

a+(b.c)



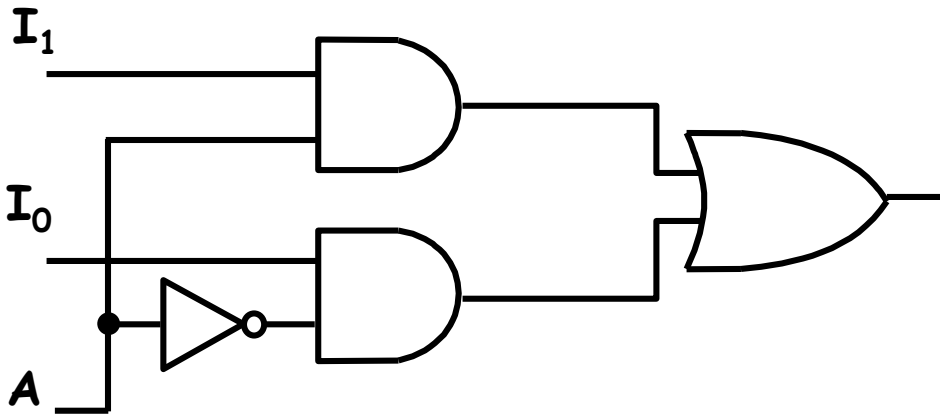
Esempi

$$(((a)' + b) \cdot c)'$$



Selettore a 2 vie

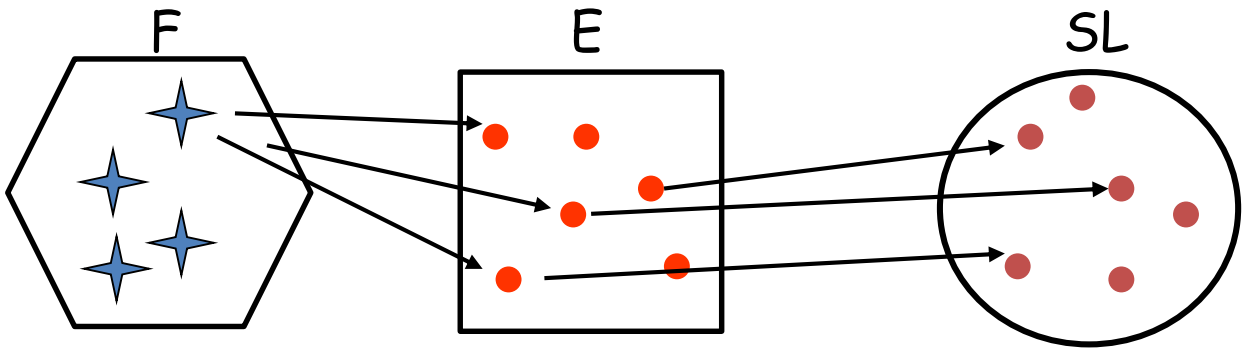
$$A' \cdot I_0 + A \cdot I_1$$



N.B. - Lo schema logico di una espressione non può avere segnali in retroazione (l'uscita di ogni gate dipende da segnali d'ingresso e/o da uscite di gate disposti "a monte").

Sintesi combinatoria

IV) Una funzione può essere descritta da una molteplicità di espressioni



- **Problema della SINTESI**: partendo da una precisa funzione, quale scegliere tra i vari schemi logici che la descrivono?
- Uno dei metodi più semplici per passare da una funzione a una espressione (tra le possibili) riguarda l'utilizzo delle cosiddette **«espressioni canoniche»**

Funzioni e Espressioni (canoniche)

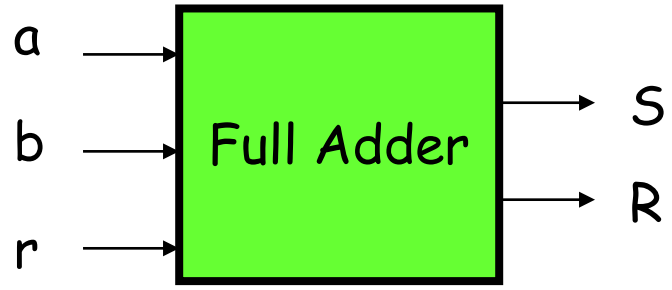
V) *Espressione canonica SP (Somma di Prodotti)*

Iª forma canonica - Ogni funzione di n variabili è descritta da una somma di tanti prodotti logici quante sono le configurazioni per cui vale 1. In ciascun prodotto, o mintermine, appare ogni variabile, in forma vera se nella configurazione corrispondente vale 1, in forma complementata se vale 0.

VI) *Espressione canonica PS (Prodotto di Somme)*

IIª forma canonica - Ogni funzione di n variabili è descritta da un prodotto di tante somme logiche quante sono le configurazioni per cui vale 0. In ciascuna somma, o maxtermine, appare ogni variabile, in forma vera se nella configurazione corrispondente vale 0, in forma complementata se vale 1.

Espressione canonica SP (1ª forma canonica)



	a	b	r	S	R
C_0	0	0	0	0	0
C_1	0	0	1	1	0
C_2	0	1	0	1	0
C_3	0	1	1	0	1
C_4	1	0	0	1	0
C_5	1	0	1	0	1
C_6	1	1	0	0	1
C_7	1	1	1	1	1

S=1 se
la configurazione d'ingresso è
C_1 o C_2 o C_4 o C_7

ovvero se
(a=0) e (b=0) e (r=1) o
(a=0) e (b=1) e (r=0) o
(a=1) e (b=0) e (r=0) o
(a=1) e (b=1) e (r=1)

ovvero se
(a'=1) e (b'=1) e (r=1) o
(a'=1) e (b=1) e (r'=1) o
(a=1) e (b'=1) e (r'=1) o
(a=1) e (b=1) e (r=1)

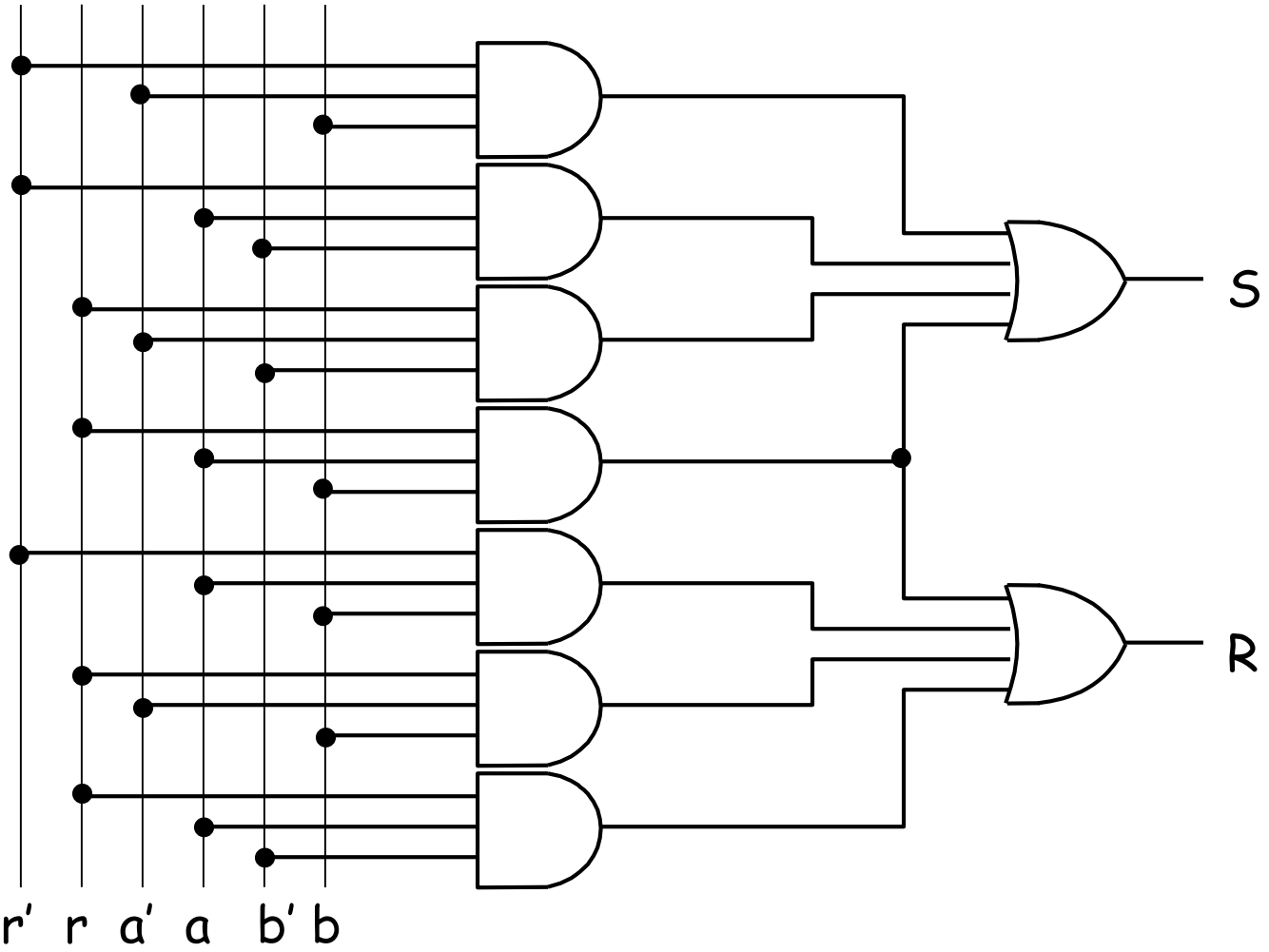
S = a'b'r + a'br' + ab'r' + abr

R = a'br + ab'r + abr' + abr

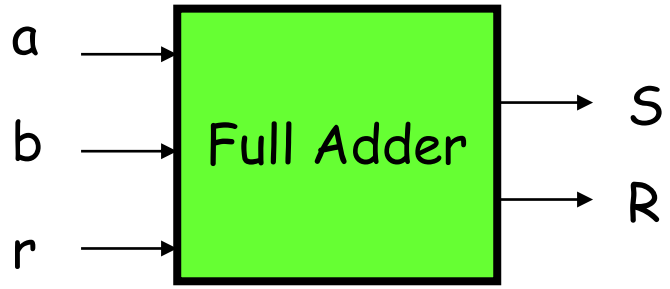
Sintesi canonica (1ª forma) del Full Adder

$S = a' b' r + a' b r' + a b' r' + a b r$

$R = a' b r + a b' r + a b r' + a b r$



Espressione canonica PS (2ª forma canonica)



	a	b	r	S	R
C_0	0	0	0	0	0
C_1	0	0	1	1	0
C_2	0	1	0	1	0
C_3	0	1	1	0	1
C_4	1	0	0	1	0
C_5	1	0	1	0	1
C_6	1	1	0	0	1
C_7	1	1	1	1	1

S=1 se
la configurazione d'ingresso è
non C_0 e non C_3 e non C_5 e non C_6

ovvero se
((a=1) o (b=1) o (r=1)) e
((a=1) o (b=0) o (r=0)) e
((a=0) o (b=1) o (r=0)) e
((a=0) o (b=0) o (r=1))

ovvero se
((a=1) o (b=1) o (r=1)) e
((a=1) o (b'=1) o (r'=1)) e
((a'=1) o (b=1) o (r'=1)) e
((a'=1) o (b'=1) o (r=1))

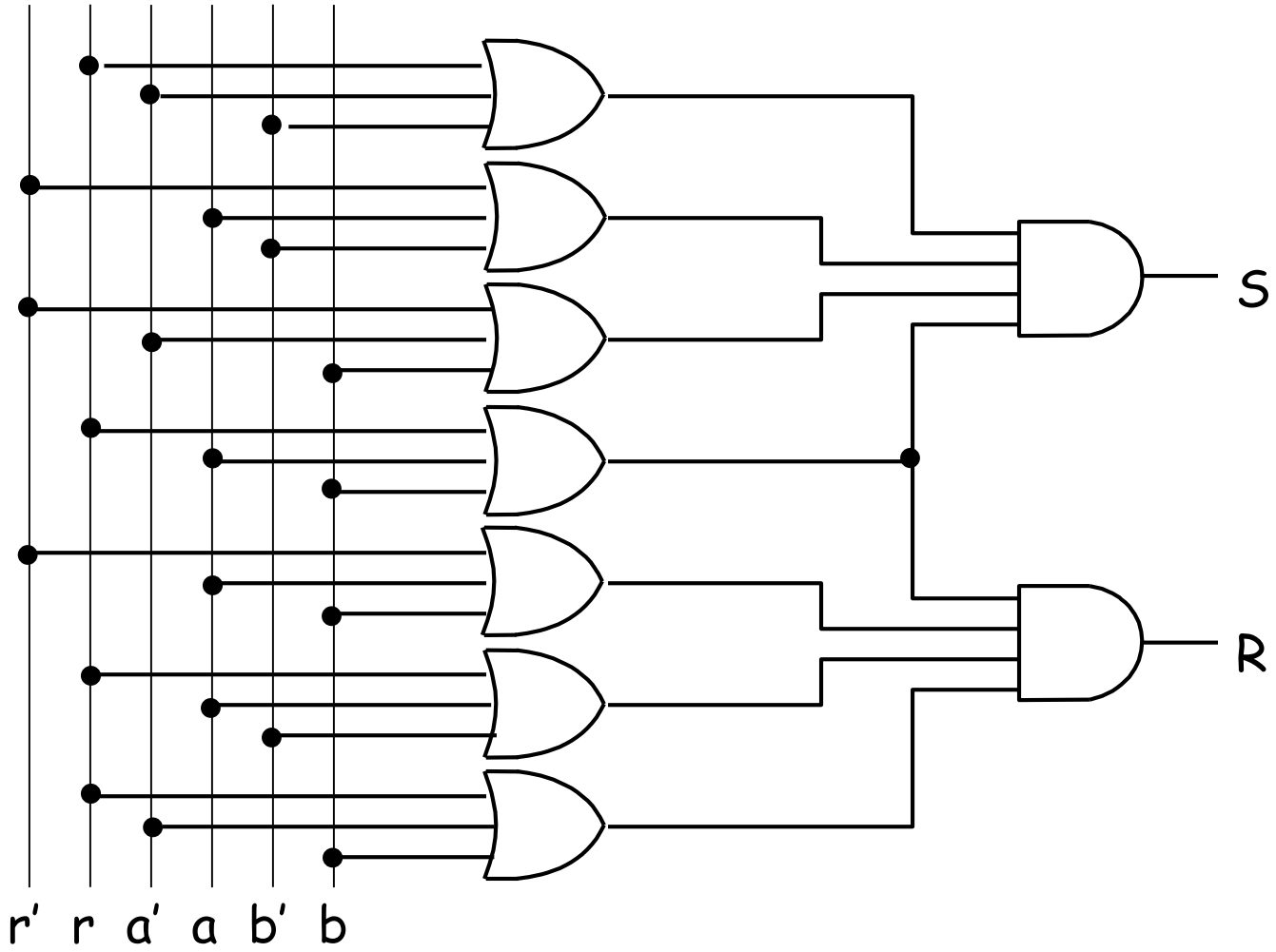
S = (a+b+r) (a+b'+r') (a'+b+r') (a'+b'+r)

R = (a+b+r) (a+b+r') (a+b'+r) (a'+b+r)

Sintesi canonica (2ª forma) del Full Adder

$S = (a+b+r)\,(a+b'+r')\,(a'+b+r')\,(a'+b'+r)$

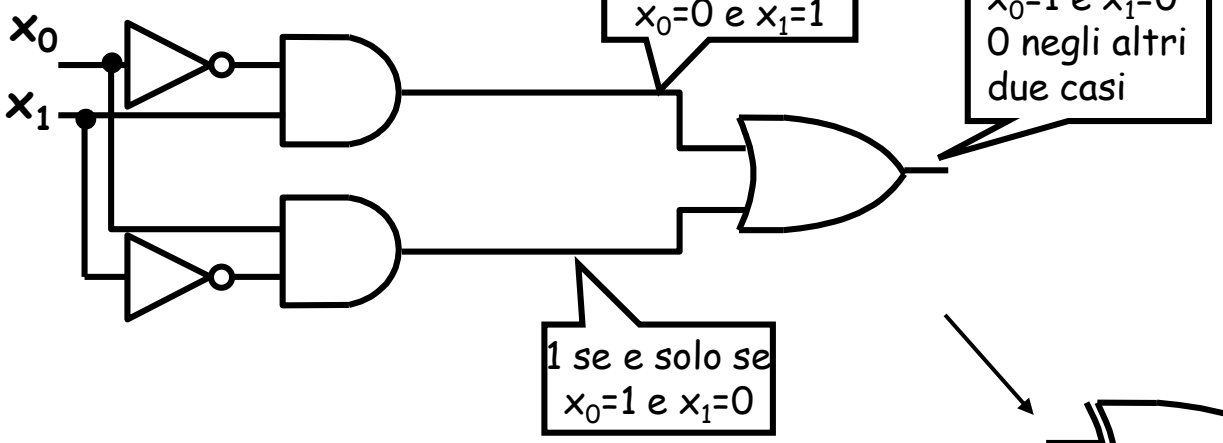
$R = (a+b+r)\,(a+b+r')\,(a+b'+r)\,(a'+b+r)$



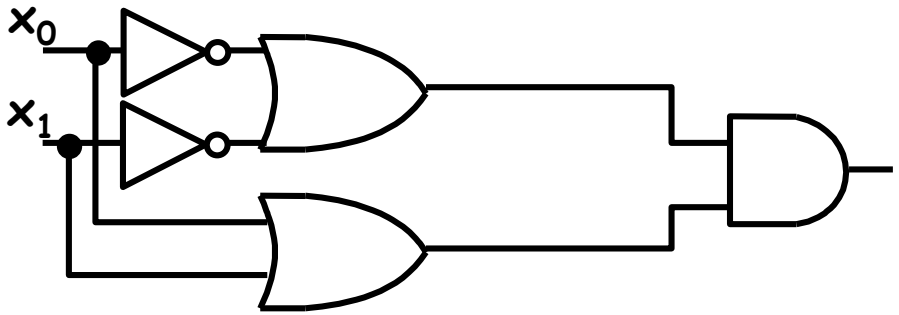
Sintesi canonica del EX-OR

I^a forma canonica (SP):
$F(x_0, x_1) = x_0' x_1 + x_0 x_1'$

x_0	x_1	$x_0 \oplus x_1$
0	0	0
0	1	1
1	0	1
1	1	0

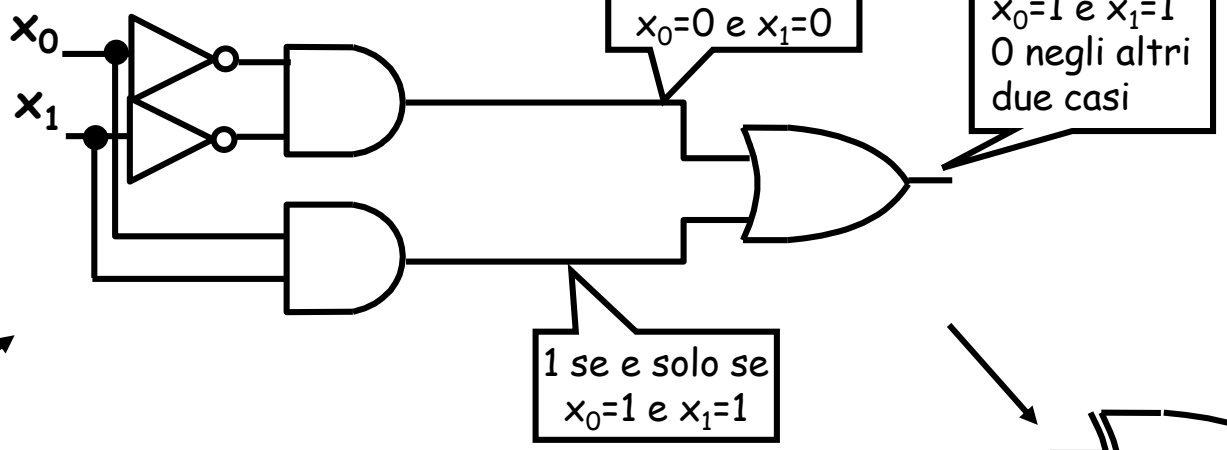


II^a forma canonica (PS):
$F(x_0, x_1) = (x_0 + x_1)(x_0' + x_1')$

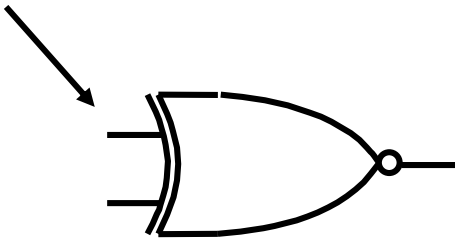


Sintesi canonica dell'Equivalence

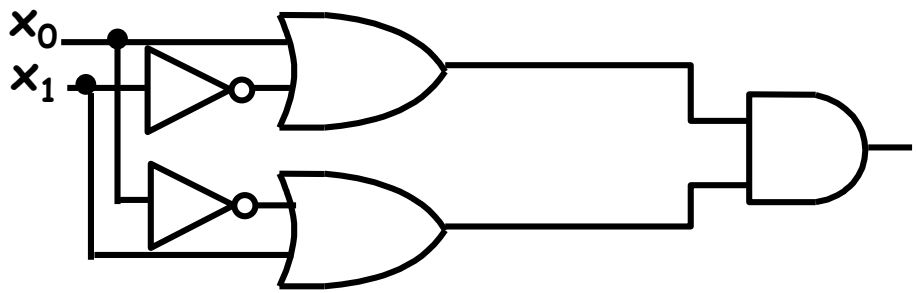
Iª forma canonica (SP):
$F(x_0, x_1) = x_0' x_1' + x_0 x_1$



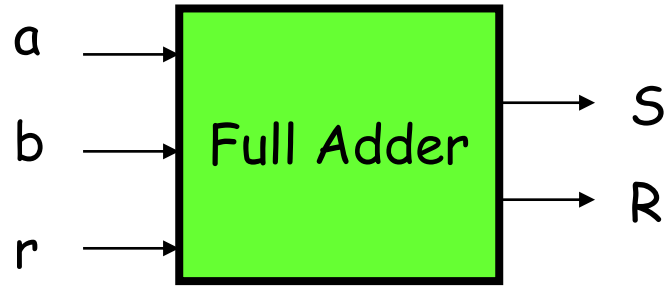
x_0 x_1	$x_0 \equiv x_1$
0 0	1
0 1	0
1 0	0
1 1	1



IIª forma canonica (PS):
$F(x_0, x_1) = (x_0 + x_1')(x_0' + x_1)$



Espressioni canoniche: notazione simbolica



$S(a,b,r) = \Sigma_3\, m(1,2,4,7)$
$= \Pi_3\, M(0,3,5,6)$

$R(a,b,r) = \Sigma_3\, m(3,5,6,7)$
$= \Pi_3\, M(0,1,2,4)$

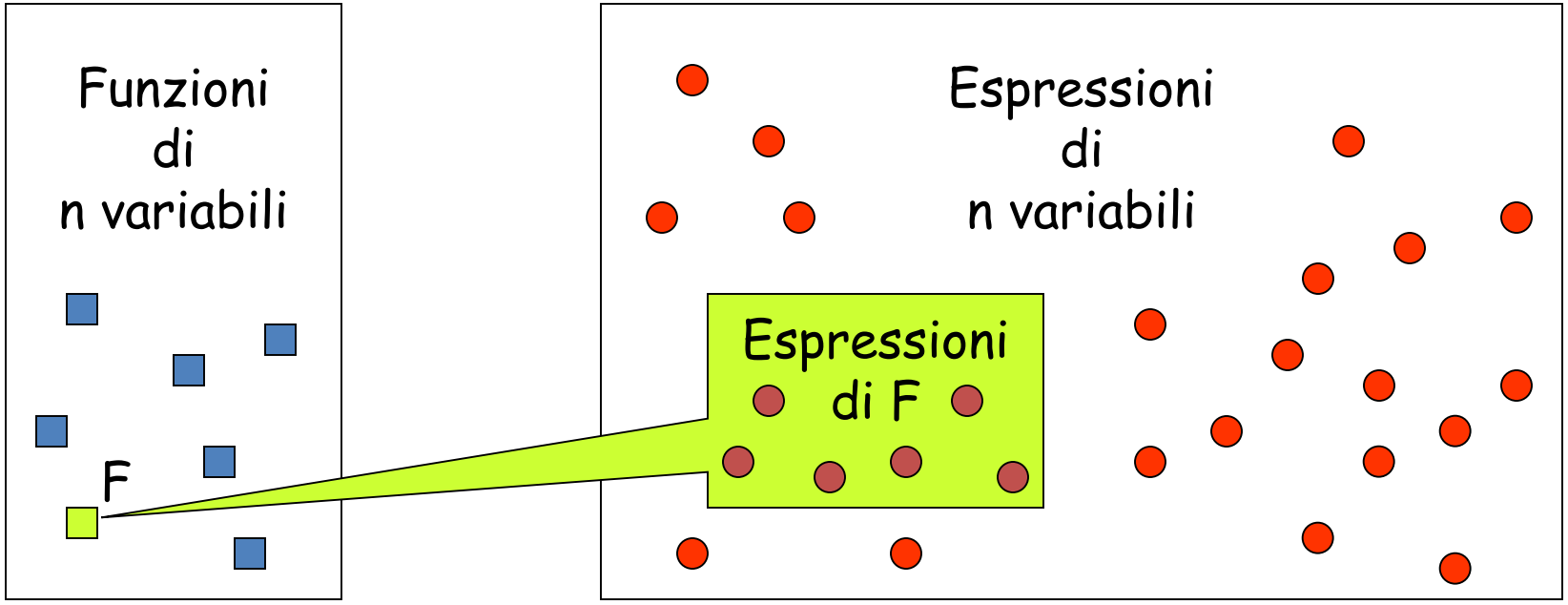
i	a	b	r	S	R
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

- m(i) : mintermine di n bit che assume il valore 1 solo per la n-pla di valori delle variabili corrispondente all'indice i
- M(i) : maxtermine di n bit che assume il valore 0 solo per la n-pla di valori delle variabili corrispondente all'indice i

- Pedice dell'operatore Σ / Π : numero di variabili coinvolte nei mintermini/maxtermini

Equivalenza tra espressioni

Espressioni equivalenti - Due espressioni E_1, E_2 sono equivalenti, e si scrive $E_1 = E_2$, se e solo se descrivono la stessa funzione.



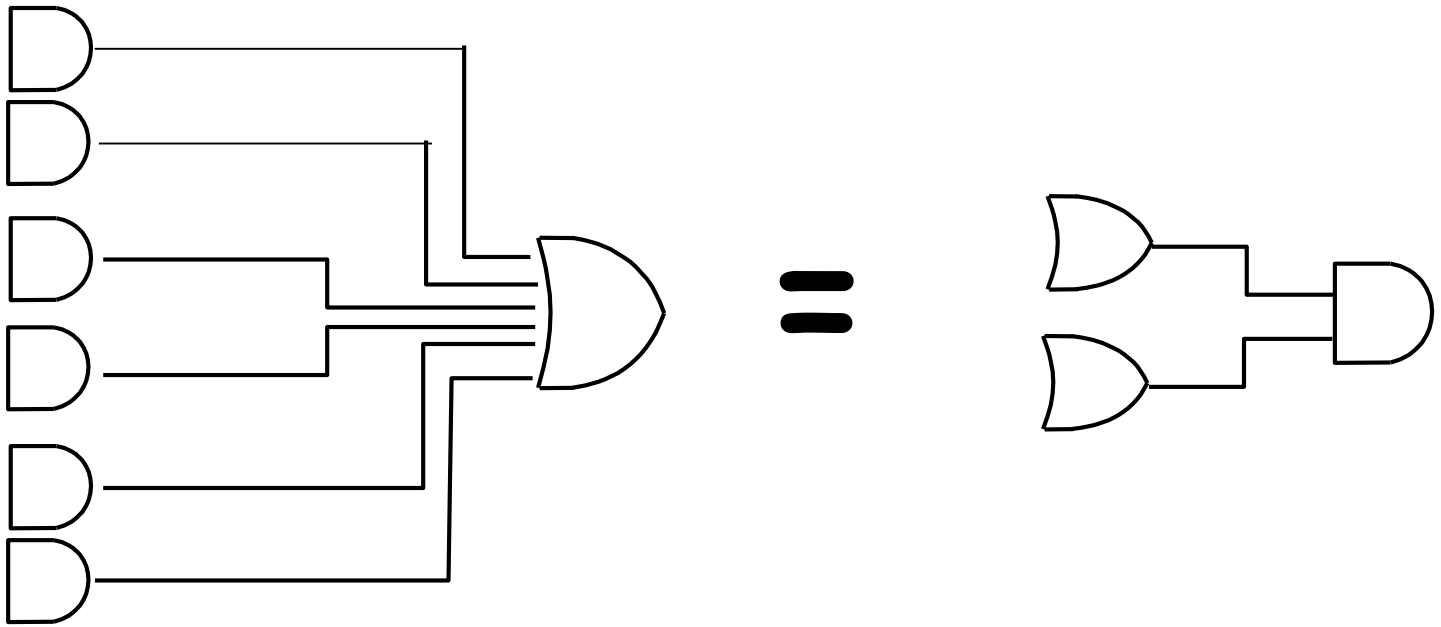
- Un esempio di espressioni equivalenti sono le due espressioni canoniche (forma PS e SP) appena viste

Equivalenza tra espressioni

- Espressioni equivalenti possono avere complessità algebrica differente, a cui corrispondono schemi logici di **complessità differente**
- Esempio: funzione U di 3 variabili a,b,c con 6 mintermini e 2 maxtermini:

$$U(a,b,c) = \sum_3 m(0,1,2,3,4,5) = \prod_3 M(6,7)$$

- Le sue espressioni canoniche richiedono
  - 6 AND e 1 OR nella forma SP
  - 2 OR e 1 AND nella forma PS



Espressioni di funzioni incomplete

Espressioni equivalenti di funzioni incomplete - Espressioni che forniscono eguale valutazione **limitatamente al dominio di una funzione incompleta** sono dette equivalenti.

- a seconda del valore assegnato alle configurazioni non utilizzate dalla funzione che realizza un Encoder a 3 ingressi, ottengo due coppie di espressioni diverse tra loro equivalenti
- riempiendo le configurazioni non utilizzate con degli «uni» anzichè degli «zeri» ottengo una espressione più semplice delle due uscite

ENCODER a 3 ingressi

x_3	x_2	x_1	z_1	z_0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
1	0	0	1	1

N.B.: le altre configurazioni sono per ipotesi impossibili

x_3	x_2	x_1	z_1	z_0	u_1	u_0
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	1	0	1	0
1	0	0	1	1	1	1
0	1	1	0	0	1	1
1	0	1	0	0	1	1
1	1	0	0	0	1	1
1	1	1	0	0	1	1

$z_1 = x_3x_2'x_1' + x_3'x_2x_1'$ $\quad u_1 = x_3 + x_2$

$z_0 = x_3x_2'x_1' + x_3'x_2'x_1$ $\quad u_0 = x_3 + x_1$

Sintesi di reti combinatorie

- L'approccio che seguiremo per procedere, assegnata una funzione di partenza, nel processo di sintesi di una rete combinatoria è il seguente:

  1) Scegliere una espressione tra le molteplici corrispondenti a una data funzione (es. le espressioni canoniche)

  2) Operare nel dominio delle espressioni per ridurne la complessità algebrica mediante manipolazione algebrica sfruttando le **equivalenze notevoli** dell'algebra di commutazione

Equivalenze notevoli

Proprietà della somma e del prodotto logico:

E1) *commutativa*

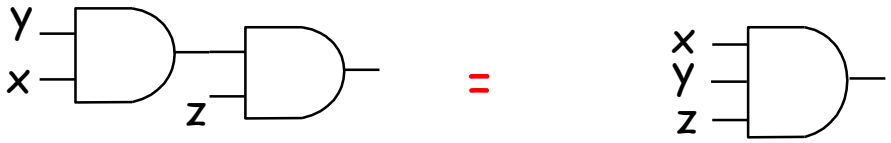
$x + y = y + x$

$x \cdot y = y \cdot x$

E2) *associativa*

$(x + y) + z = x + y + z$

$(x \cdot y) \cdot z = x \cdot y \cdot z$

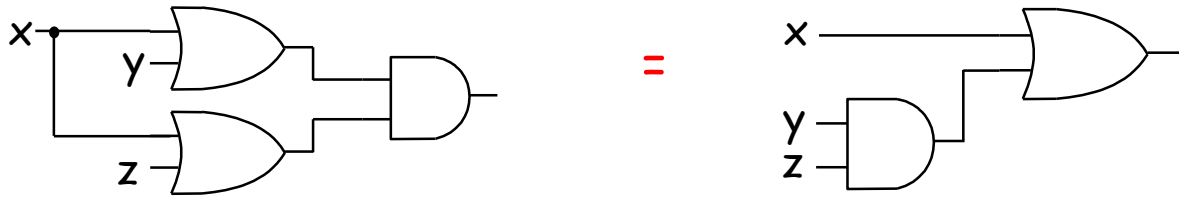


(utile per ridurre il fan-in)

Equivalenze notevoli

E3) *distributiva*

$$(x \cdot y) + (x \cdot z) = x \cdot (y + z)$$
$$(x + y) \cdot (x + z) = x + (y \cdot z)$$



(riduzione del numero di gate utilizzati)

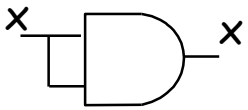
x	y	z	a=x·y	b=x·z	a+b	c=y+z	x·c
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	0
0	1	1	0	0	0	1	0
1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

Equivalenze notevoli

E4) *idempotenza*

$x + x = x$

$x \cdot x = x$



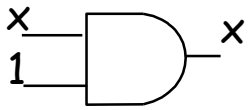
x	x·x
0	0
1	1

(amplificazione in presenza di gate di un solo tipo?)

E5) *identità*

$x + 0 = x$

$x \cdot 1 = x$

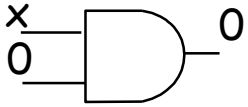


x	x·1
0	0
1	1

E6) *limite*

$x + 1 = 1$

$x \cdot 0 = 0$

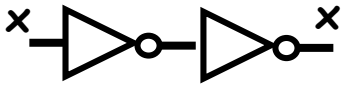


x	x·0
0	0
1	0

Equivalenze notevoli

Proprietà della complementazione:

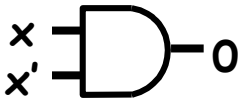
E7) *involuzione* $(x')' = x$



(amplificazione di segnale)

E8) *limitazione* $x + x' = 1$

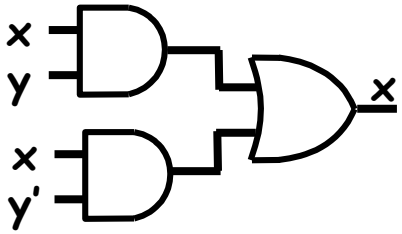
$x \cdot x' = 0$



x	x'	x·x'
0	1	0
1	0	0

E9) *combinazione* $xy + xy' = x$

$(x+y)\cdot(x+y') = x$



Dim.:

$xy + xy' = x\,(y+y')$ (E3 – distributiva)

$= x\cdot 1$ (E8 – limitazione)

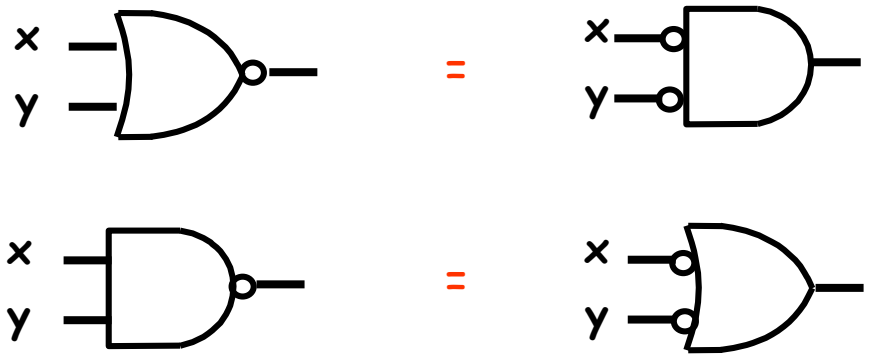
$= x$ (E5 – identità)

(prorietà fondamentale ai fini della semplificazione algebrica delle espressioni – lo si analizzerà in seguito)

Equivalenze notevoli

E10) *I^a legge di De Morgan* $(x + y)' = x' \cdot y'$

II^a legge di De Morgan $(x \cdot y)' = x' + y'$



x	y	(x·y)'	x'	y'	x'+y'
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

- utile per determinare espressioni equivalenti utilizzando gate logici diversi, es. NOR/NAND al posto di AND/OR
- Equivalentemente, le leggi di De Morgan asseriscono che:

  $x+y = ((x+y)')'$ E7-involuzione

  $= (x' \cdot y')'$ E10-De Morgan
- Dunque è possibile sostituire un gate OR con un gate AND (e viceversa)
  1. Negando l'uscita
  2. Negando tutti gli ingressi

Equivalenze notevoli

E11) *consenso* $xy + x'z + yz = xy + x'z$

$(x+y)\cdot(x'+z)\cdot(y+z) = (x+y)\cdot(x'+z)$

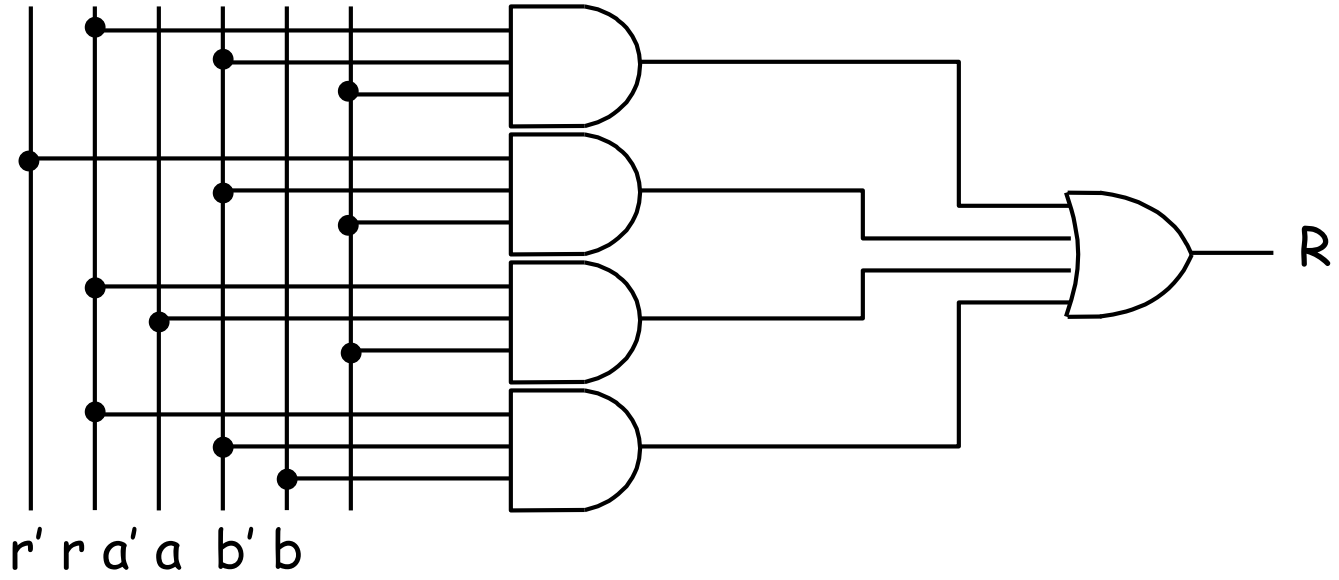
Dimostrazione:

$xy + x'z + yz$	$= xy + x'z + yz\,(x + x')$	Limitazione e Identità
	$= xy + x'z + yzx + yzx'$	Distributiva
	$= xy\,(1+z) + x'z\,(1+y)$	Distributiva
	$= xy\,(1) + x'z\,(1)$	Limite
	$= xy + x'z$	Identità

Manipolazione algebrica di espressioni ...

- Lo schema circuitale relativo all'espressione canonica SP del «riporto» (R) di un Full Adder richiede 4 «AND» a 3 ingressi ciascuno e, in cascata, 1 «OR» a 4 ingressi (ipotizzando di avere a disposizione ogni ingresso in forma vera e complementata)

$$R = a' b r + a b' r + a b r' + a b r$$



... Manipolazione algebrica di espressioni ...

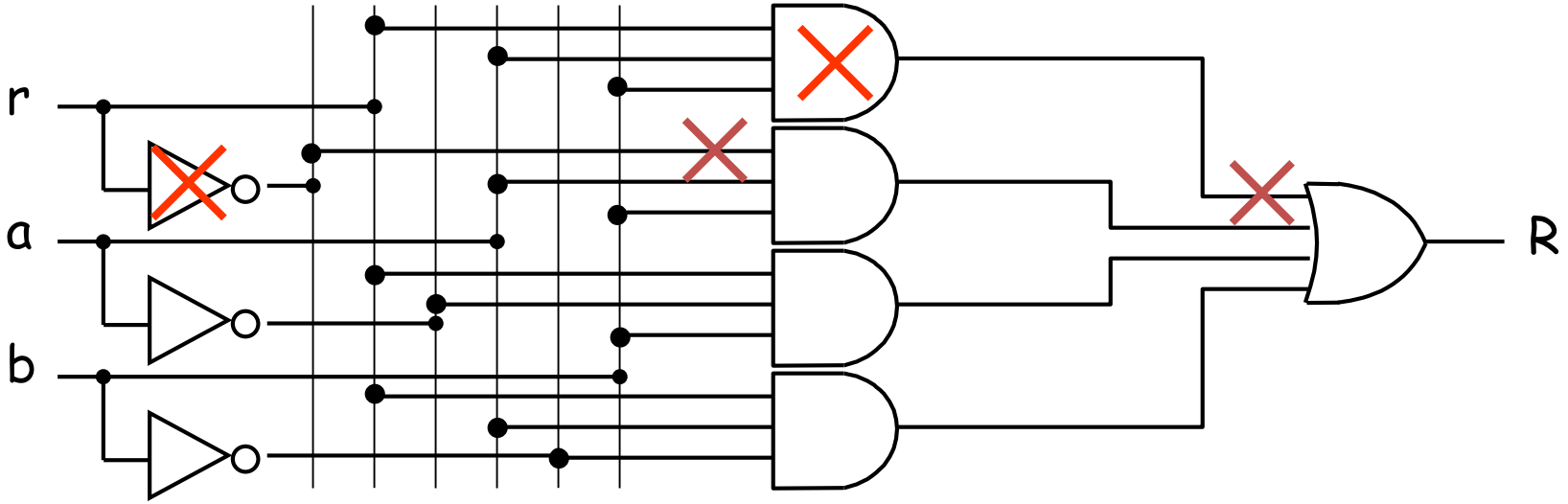
- Proviamo a manipolare algebricamente l'espressione SP ai fini di ottenere uno schema logico semplificato

$$R = a' b r + a b' r + a b r' + a b r$$

$$= a' b r + a b' r + a b (r' + r) \qquad \text{Distrib. } (E_3)$$

$$= a' b r + a b' r + a b 1 \qquad \text{Limitazione } (E_8)$$

$$= a' b r + a b' r + a b \qquad \text{Identità } (E_5)$$



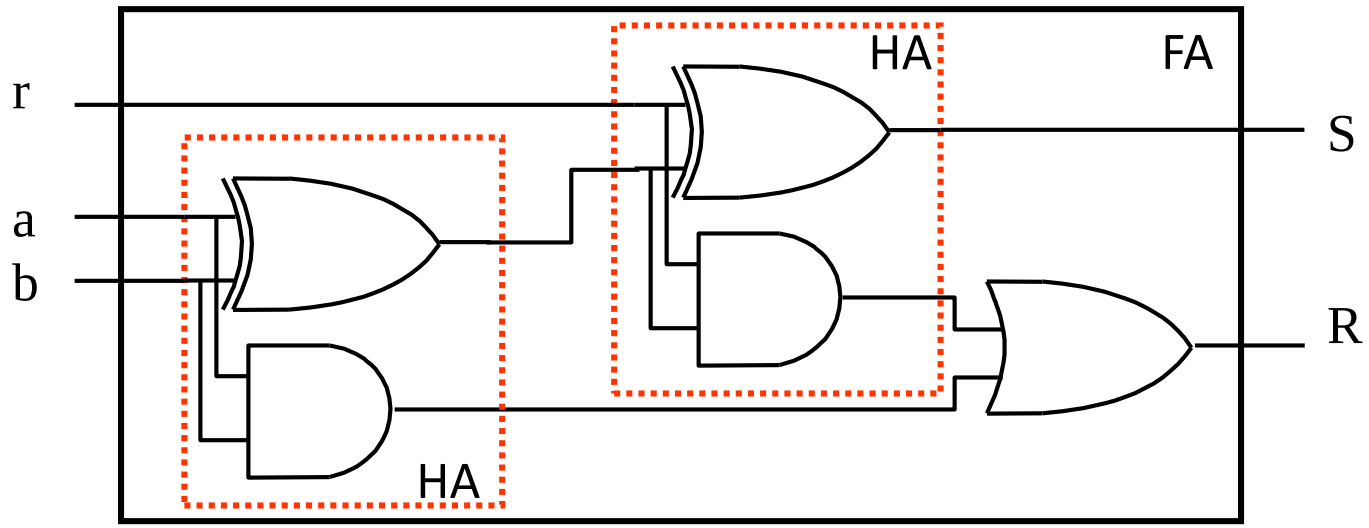
... Manipolazione algebrica di espressioni ...

- Sfruttando la definizione di Ex-or possiamo poi notare che:

$$R = a' b r + a b' r + a b$$
$$= r (a'b + ab') + ab = \mathbf{r (a \oplus b) + ab}$$

- Analogamente, partendo dall'espressione canonica SP della «somma» (S) di un Full Adder ed applicando le equivalenze notevoli:

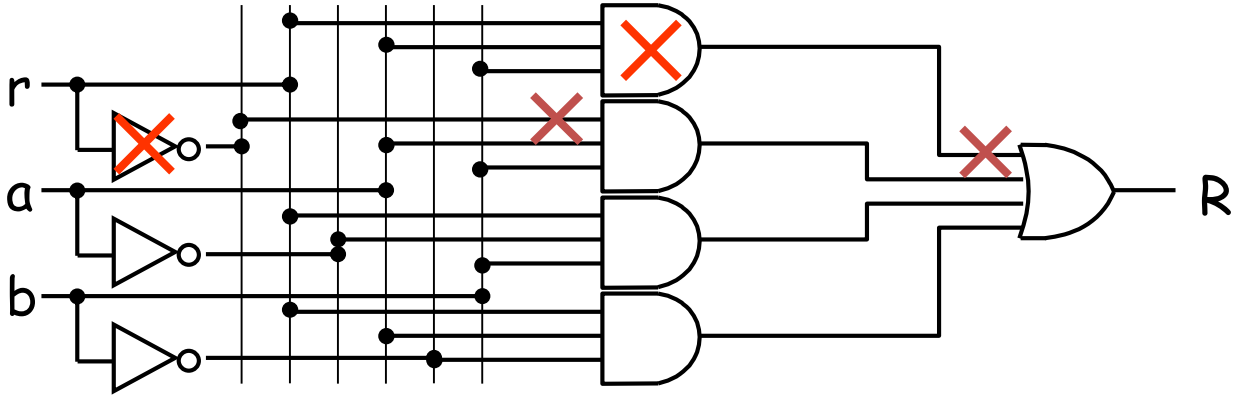
$$S = r' a' b + r' a b' + r a' b' + r a b$$
$$= r' (a' b + a b') + r (a' b' + a b) \quad \text{Distrib. } (E_3)$$
$$= r' (a \oplus b) + r (a \oplus b)' \quad \text{Def. Ex-or/Equivalence}$$
$$= \mathbf{r \oplus (a \oplus b)} \quad \text{Def. Ex-or}$$



... Manipolazione algebrica di espressioni ...

- Utilizzando solo gate AND, NOT e OR avevamo ottenuto una riduzione di 1 AND, 1 NOT e un fan-in ridotto per due ulteriori gate logici rispetto allo schema ottenuto dalla espressione canonica SP

$R = a' b r + a b' r + a b$



- È possibile ottenere una semplificazione ulteriore? Ripartiamo da capo applicando una manipolazione algebrica meno intuitiva..

Formulazione SP originale..

$R = a' b r + a b' r + a b r' + a b r + a b r + a b r$ — Idempot. (E_4)

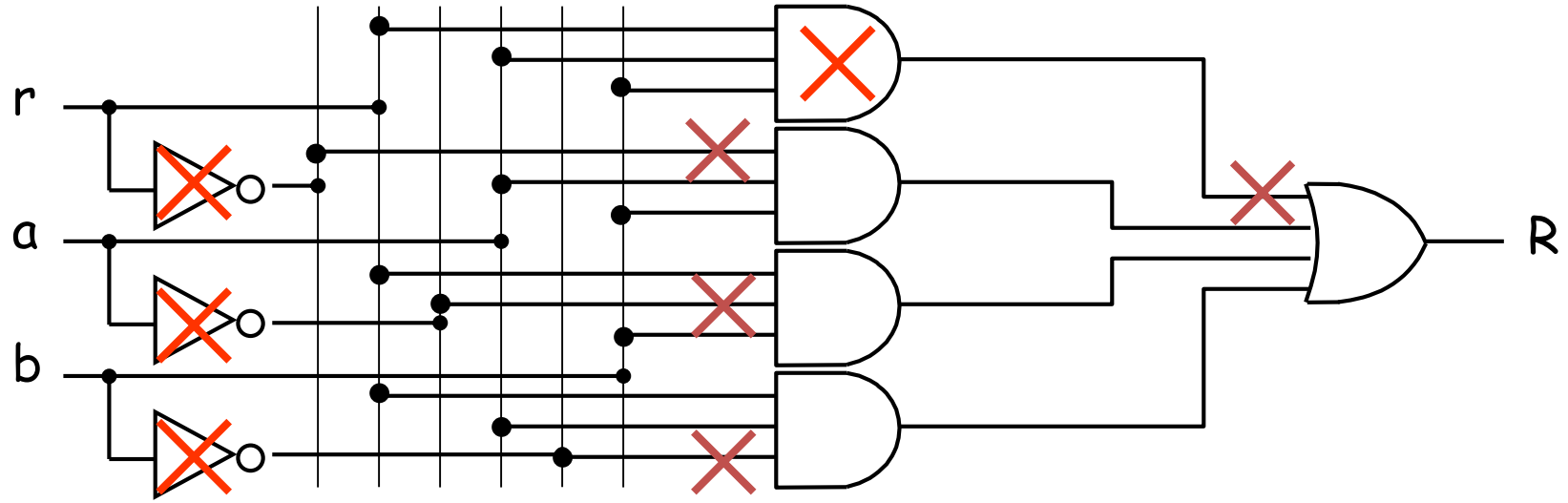
$= b r (a' + a) + a r (b' + b) + a b (r' + r)$ — Distrib. (E_3)

$= b r 1 + a r 1 + a b 1$ — Limitaz. (E_8)

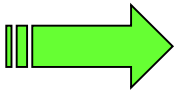
$= b r + a r + a b$ — Identità (E_5)

... Manipolazione algebrica di espressioni

$R = b\,r + a\,r + a\,b$



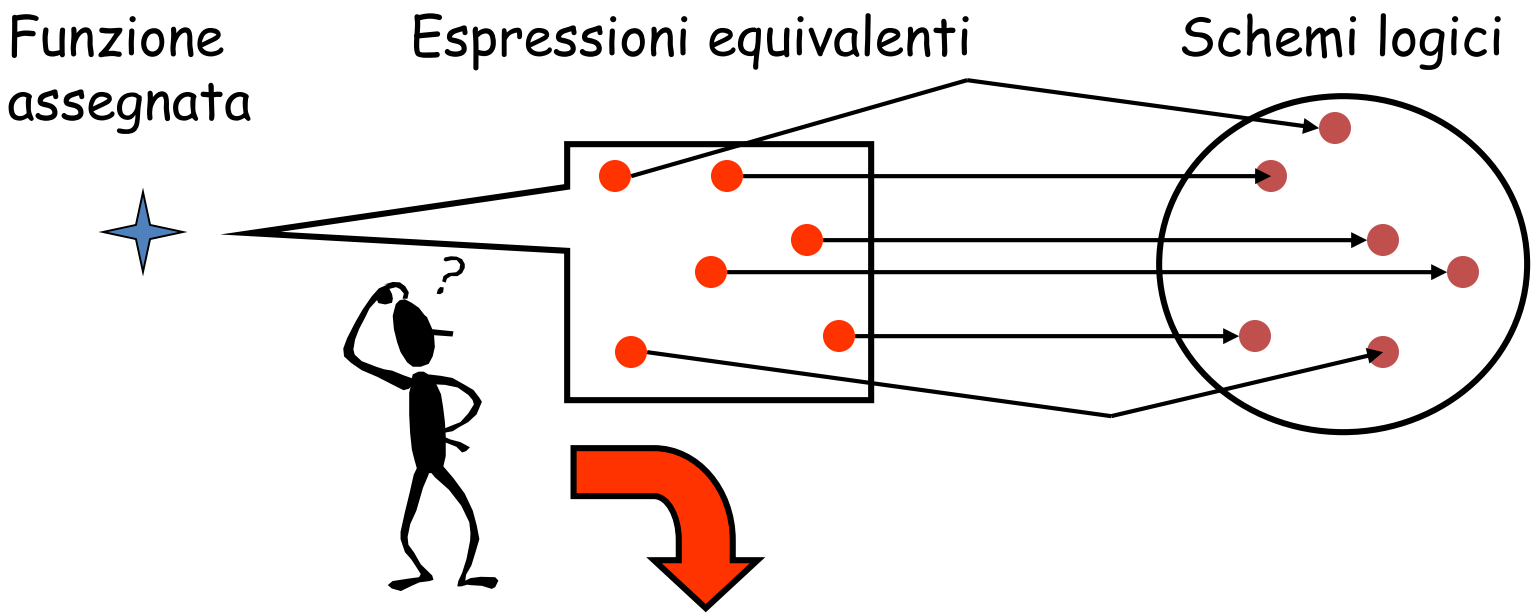
1 OR a 4 ingressi
4 AND a 3 ingressi
3 NOT



1 OR a 3 ingressi
3 AND a 2 ingressi

- Ma il procedimento di manipolazione algebrica verso la rete «*di costo minimo*», seppure a volte intuitivo, può non essere affatto banale!
- Il processo di sintesi volto all'ottenimento della rete di costo minimo viene realizzato mediante opportune metodologie (es. le mappe di Karnaugh, si vedranno prossimamente)

Il problema della sintesi



SINTESI: individuazione dell'espressione che fornisce lo schema "migliore" per la realizzazione della funzione assegnata.

- Rapidità di progetto
- Massima velocità
- Massima flessibilità
- Minima complessità

Progetto logico di circuiti combinatori