Reti Combinatorie

Reti Logiche T

Ingegneria Informatica

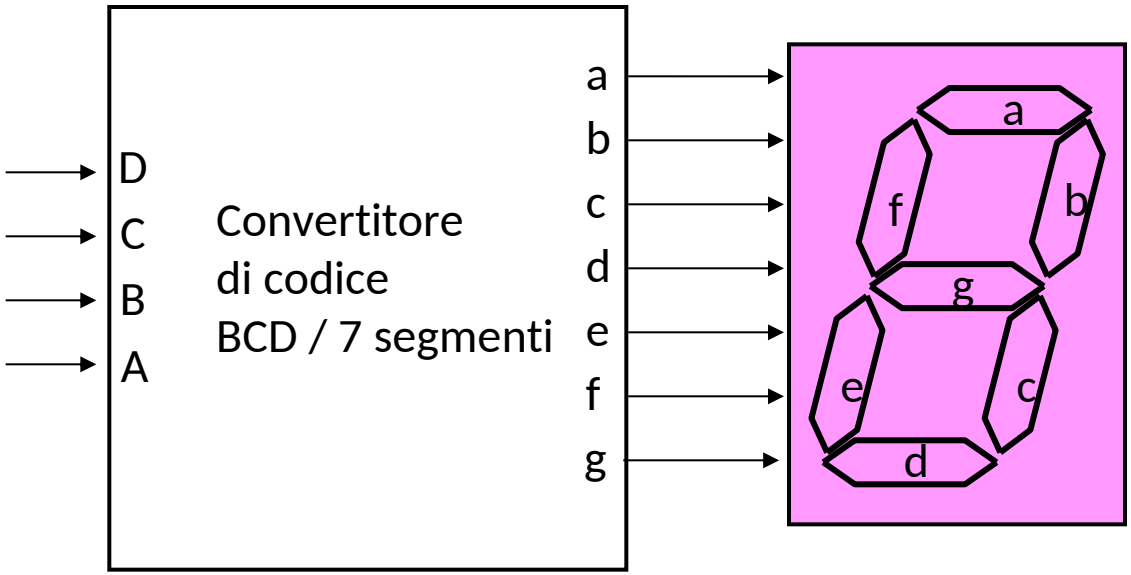
Generica rete logica

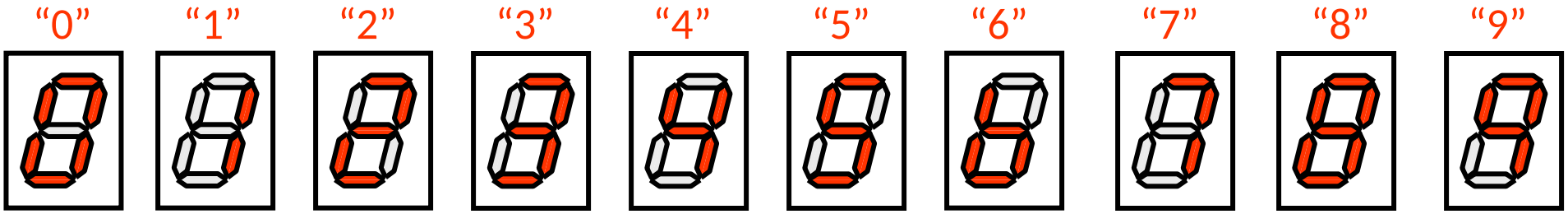
Rete logica: Modello astratto che assume come *primitive* alcune semplici elaborazioni di *segnali binari* (gate) e realizza comportamenti più complessi. Nel seguito definiremo procedimenti per stabilire

1. quale **struttura** realizza un dato **comportamento** (**sintesi**)
2. quale **comportamento** ha una data **struttura** (**analisi**)

Convenzioni: ingressi a sinistra, uscite a destra,
nomi dei segnali di ingresso/uscita della rete indicati all'interno della rete

Trascodifica BCD/7 Segmenti

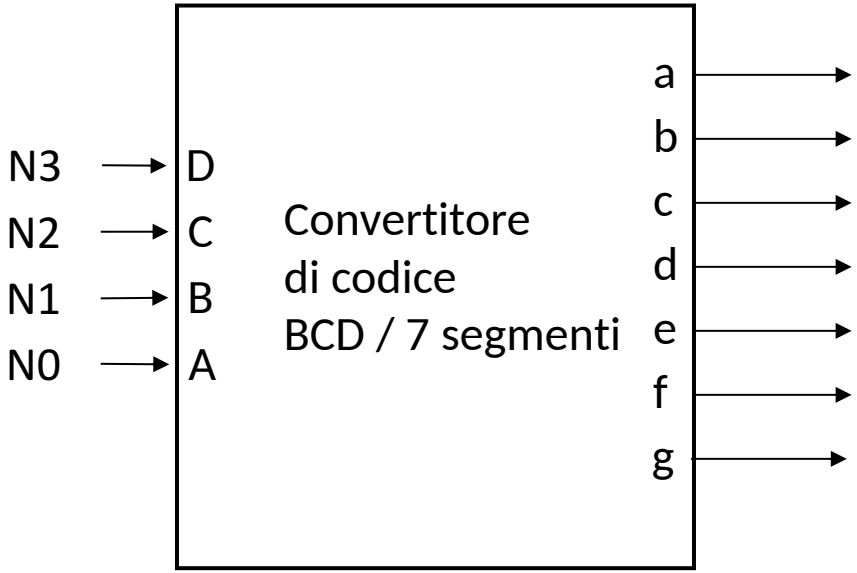




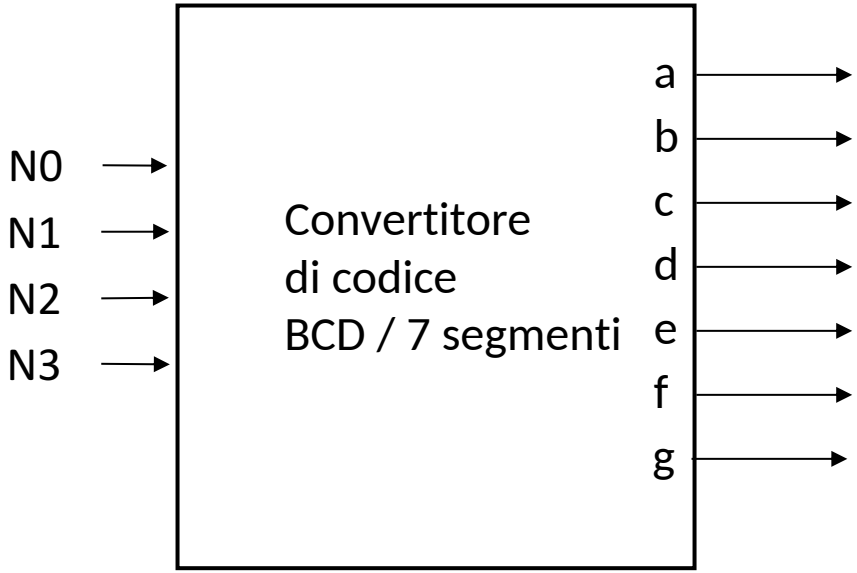
- Questo convertitore è un esempio di **macchina combinatoria**: le 7 uscite (abcdefg) sono univocamente determinate dal valore corrente dei 4 ingressi (DCBA) che codificano un numero senza segno tra 0 e 9.
- Es: DCBA=0001 -> abcdefg = 1001111

Esempi di connessione

- Dato un bus N[3..0], in cui è memorizzato un numero senza segno **con il bit più significativo in posizione 3**, come deve essere collegato al convertitore per ottenerne la rappresentazione a 7 segmenti?

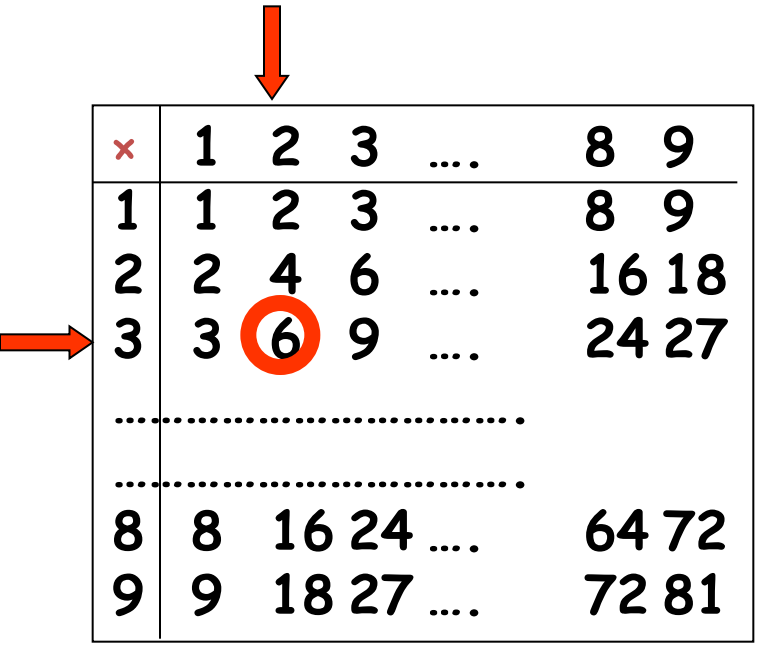


✓



?

Esempi di macchine combinatorie



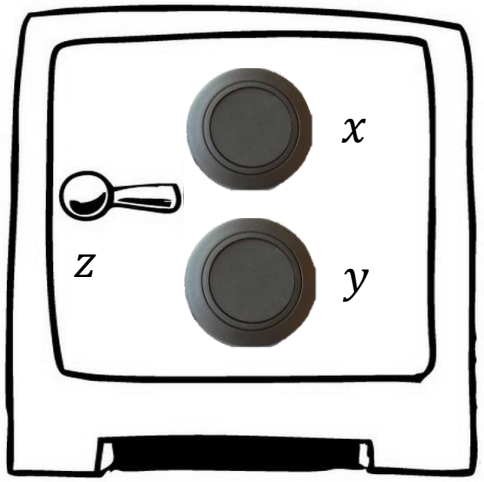
×	1	2	3	….	8	9
1	1	2	3	….	8	9
2	2	4	6	….	16	18
3	3	6	9	….	24	27
……						
……						
8	8	16	24	….	64	72
9	9	18	27	….	72	81

la tavola pitagorica

il dizionario

La cassaforte

- Macchina con 2 ingressi (x, y) e una uscita (z)
- z=0/1: cassaforte chiusa/aperta
- z=1 con ingresso 11 se e solo se la sequenza vista in ingresso è stata quella corretta, ovvero xy = 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,** è quindi una **macchina sequenziale**

Combinatoria vs. Sequenziale

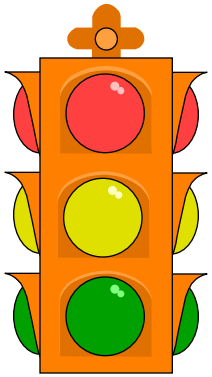
- **Rete (o macchina) combinatoria**: l'uscita dipende **unicamente dagli ingressi correnti**

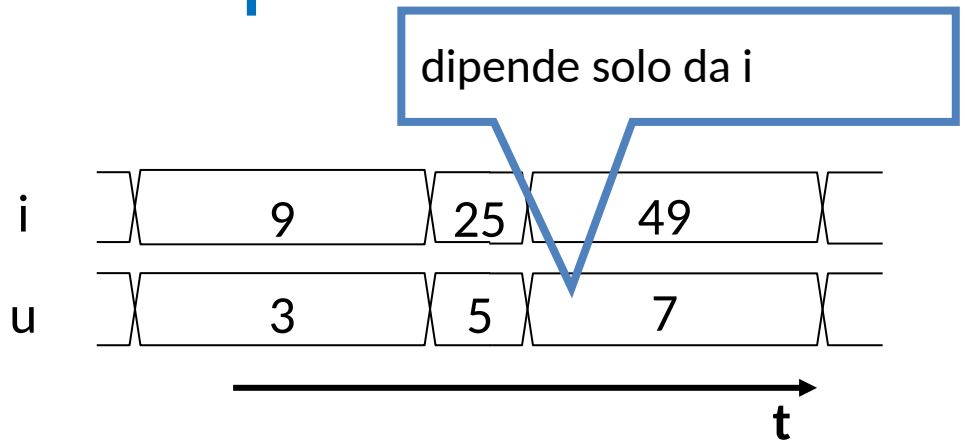
- **Rete (o macchina) sequenziale**: l'uscita non è univocamente determinata dagli ingressi correnti, ma dipende anche dalla **storia passata** (sequenza di ingressi visti in precedenza) e/o dallo **scorrere del tempo**

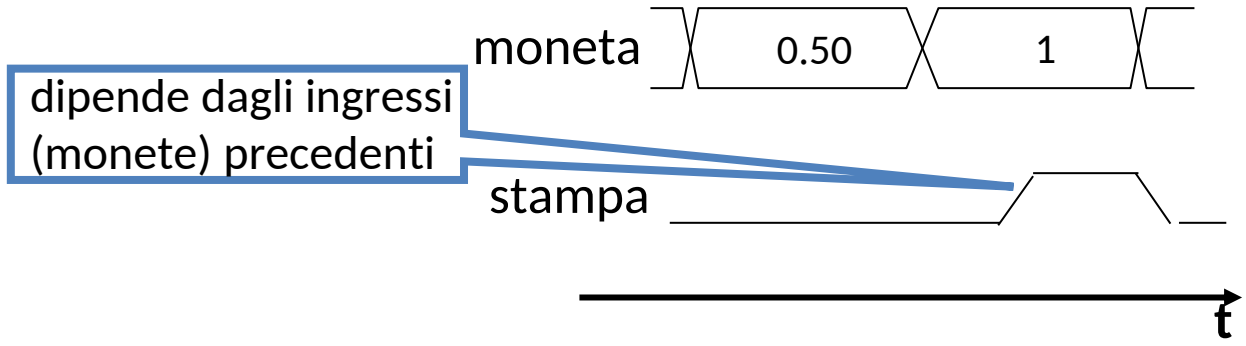
- Come faccio a capire se una rete va realizzata come combinatoria o sequenziale? **Se in presenza di una stessa configurazione di ingressi la rete deve fornire 2 o più uscite diverse**, allora è una rete sequenziale, altrimenti è una rete combinatoria

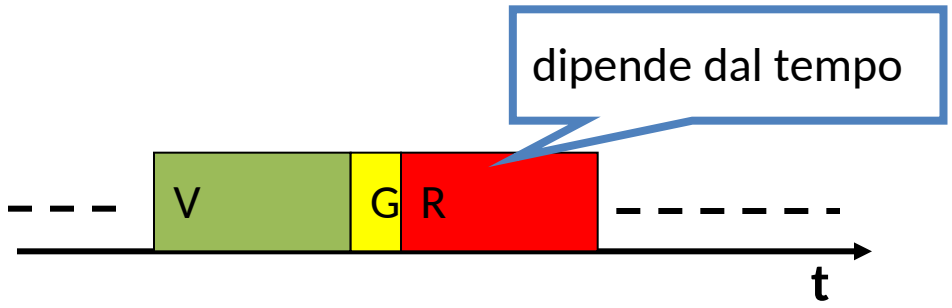
Altri esempi

$$u = \sqrt{i}$$





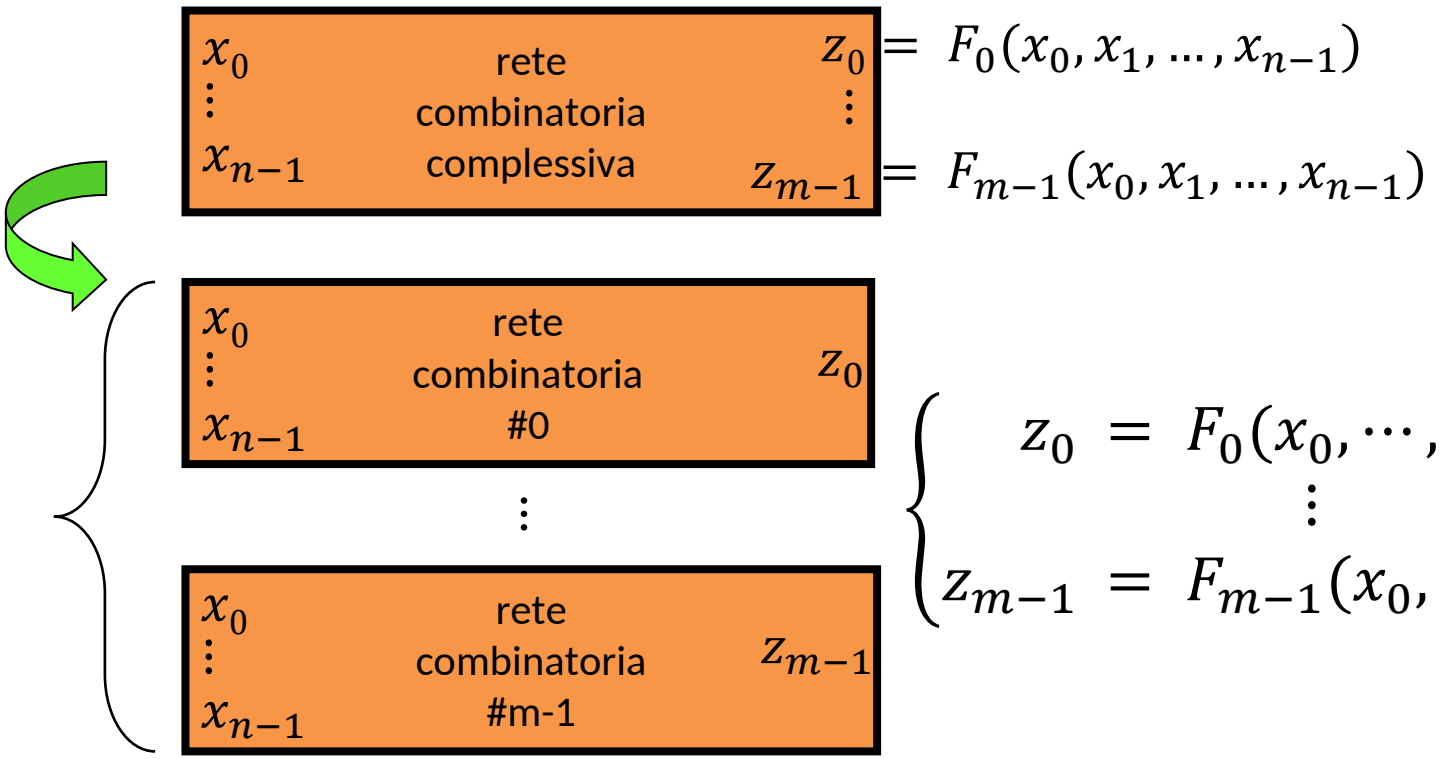




Composizione e decomposizione

La disposizione **in serie e/o in parallelo** di reti logiche combinatorie è ancora una rete logica combinatoria

Per progettare una rete con m uscite si possono quindi progettare m reti con una sola uscita, operanti in parallelo



Comportamento & Struttura

Comportamento
(cosa fa la rete)

sintesi

analisi

Struttura
(come è realizzata la rete)

Descrizione in linguaggio naturale
«la rete ha uscita z=1 quando x è uguale a y e diverso da w»

Tabella della verità

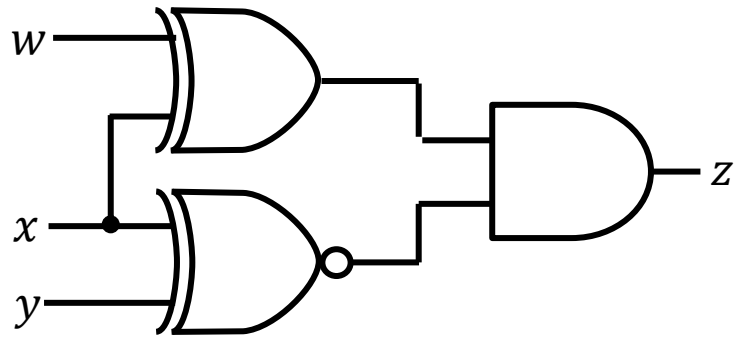
w	x	y	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
…	…	…	…

Espressione

$$z = (x \equiv y)(x \oplus w)$$

o, in modo equivalente,

Schema logico



Funzioni complete e incomplete

Funzione completa di n variabili binarie $z = F(x_0, x_1, \dots, x_{n-1})$:
per ognuna delle 2^n configurazioni degli ingressi, è definito il valore dell'uscita z.

Esempi: decoder, sommatore, selettore a n vie (multiplexer)

Funzione incompleta o non completamente specificata:
vi è almeno una configurazione degli ingressi per cui non è specificato il valore dell'uscita, o perché tali configurazioni non possono presentarsi o perché non interessa il valore dell'uscita nel caso in cui si presentino

Esempi: convertitore BCD/7 segmenti, encoder

Tabella della verità

- Descrive univocamente il comportamento di una rete combinatoria a n ingressi (ovvero di una funzione F di n variabili binarie $x_0, x_1, \dots, x_{n-1}$)
- Nel caso di funzioni non completamente specificate si possono avere
  - Meno righe, evitando di riportare le configurazioni di input che non possono presentarsi
  - Il simbolo – (*don't care*) nelle uscite (invece di 0 o 1) ad indicare una condizione di indifferenza
- NOTA BENE: **Una struttura che realizza un comportamento è sempre completamente specificata**, ovvero l'uscita avrà valore 0 o 1 per ogni configurazione di ingresso. La presenza di un *don't care* nella specifica del comportamento da al progettista la libertà di scegliere se essa varrà 0 o 1 per quella configurazione.

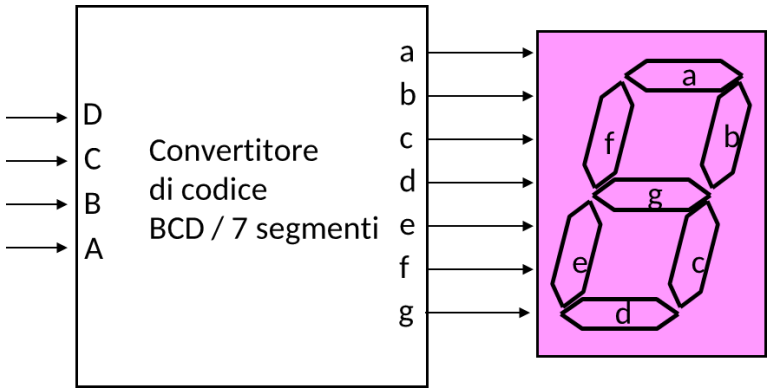
$n+1$ colonne

2^n righe (se completamente specificata)

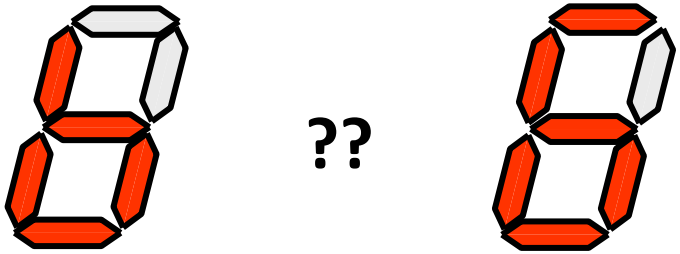
x_0	x_1	…	x_{n-1}	$F(x_0, x_1, \dots, x_{n-1})$
0	0	…	0	0 oppure 1 oppure –
0	0	…	1	0 oppure 1 oppure –
⋮	⋮	⋱	⋮	⋮
1	1	…	0	0 oppure 1 oppure –
1	1	…	1	0 oppure 1 oppure –

Il convertitore BCD/7 Segmenti

- Sono in realtà 7 funzioni in parallelo di 4 ingressi
  - a(D,C,B,A)
  - b(D,C,B,A)
  - …
  - g(D,C,B,A)



- Descrizione in linguaggio naturale non è univoca: «6» a cosa equivale?

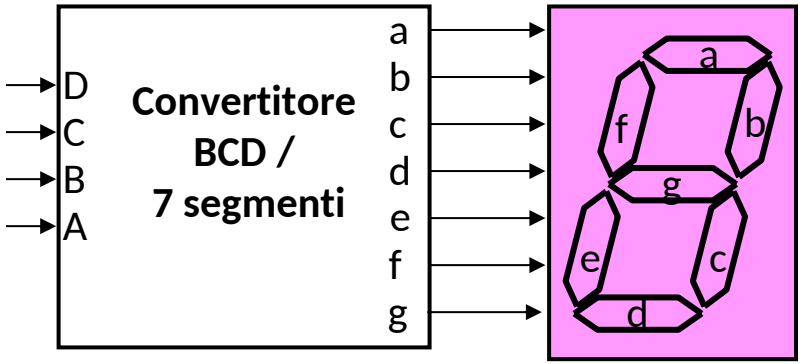


Solo la tabella della verità ci permette di descrivere in modo non ambiguo il comportamento di una macchina combinatoria

Convertitore BCD/7 Segmenti

7 funzioni di 4 variabili

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



Segmento attivo se segnale «0» (uscite attive basse), quindi, per esempio, 8 -> uscite tutte «0»

Le configurazioni che non possono presentarsi in input possono non essere riportate oppure utilizzo per l'uscita il simbolo che rappresenta una **condizione di «indifferenza»** (*don't care*)

Funzioni di n variabili

- Abbiamo già enumerato tutte le funzioni di una e due variabili binarie
- Non potremmo fare lo stesso con 4 ingressi, e scegliere tra queste i gate a 4 ingressi che realizzano le 7 funzioni del convertitore a 7 segmenti?

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

4 funzioni di una variabile

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

16 funzioni di due variabili

Funzioni complete di n variabili

Il numero di distinte funzioni di n variabili binarie

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

aumenta esponenzialmente con n

n	$\Phi(n)$
1	4
2	16
3	256
4	65536
…	…

I costruttori non forniscono la realizzazione ad hoc di ogni funzione

E' il progettista che deve realizzarle tramite opportuna composizione di gate elementari disponibili in forma di circuiti integrati

- Small Scale Integration (SSI)
- Medium SI (MSI)
- Large SI,Very LSI (LSI,VLSI)

Sintesi: da comportamento a struttura

Comportamento
(cosa fa la rete)

sintesi

analisi

Descrizione in linguaggio naturale
«la rete ha uscita z=1 quando x è uguale a y e diverso da w»

Tabella della verità

w	x	y	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
…	…	…	…

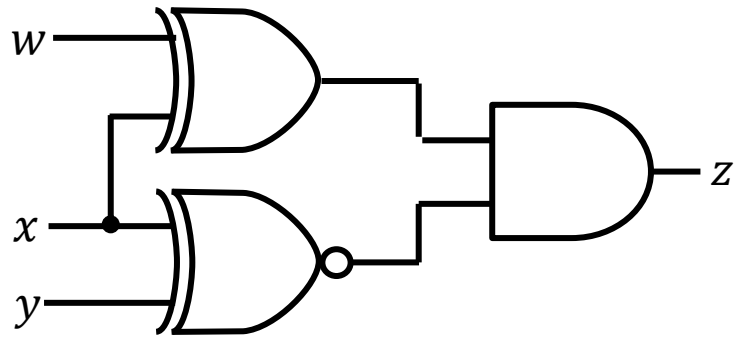
Struttura
(come è realizzata la rete)

Espressione

$$z = (x \equiv y)(x \oplus w)$$

o, in modo equivalente,

Schema logico



Algebre binarie

Algebra binaria: Sistema matematico formato da un insieme di operatori definiti assiomaticamente ed atti a descrivere con una espressione ogni possibile 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 sono sufficienti per esprimere ogni possibile funzione binaria

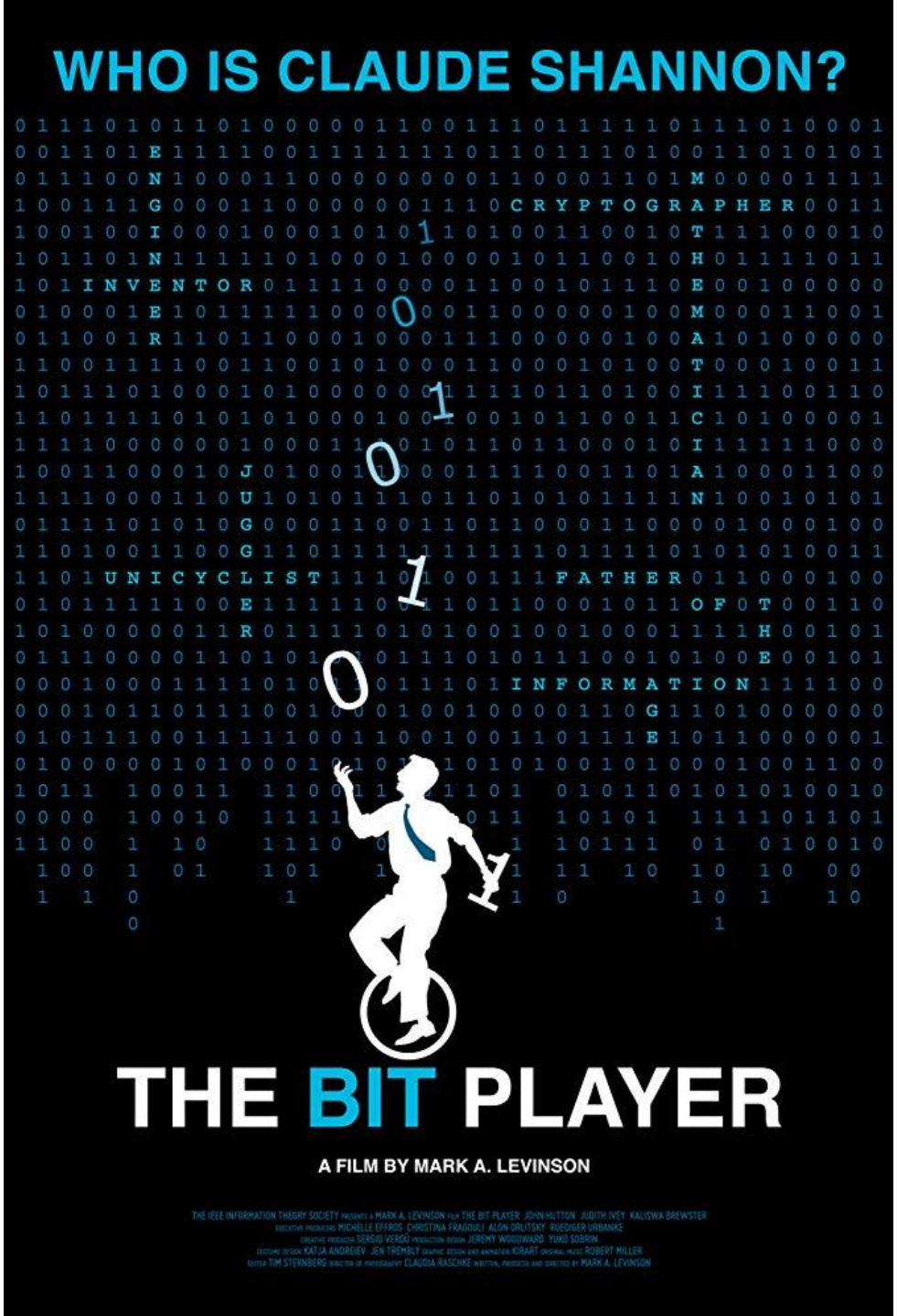
Algebra di commutazione
{0, 1} {+, . , ’}
tre operatori

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)


WHO IS CLAUDE SHANNON?
THE BIT PLAYER
A FILM BY MARK A. LEVINSON

Algebra di commutazione

Algebra binaria definita da

1. un insieme di **simboli** $B = \{0, 1\}$
2. un insieme di **operazioni** $O = \{+, ., '\}$, ovvero

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

3. un insieme di **postulati** P:

$$0 + 0 = 0 \qquad 0 . 0 = 0 \qquad 0' = 1$$

$$1 + 0 = 1 \qquad 1 . 0 = 0 \qquad 1' = 0$$

$$0 + 1 = 1 \qquad 0 . 1 = 0$$

$$1 + 1 = 1 \qquad 1 . 1 = 1$$

N.B. somma logica, non aritmetica

I postulati corrispondono al comportamento dei gate OR, AND e NOT

Cos'è un'espressione?

Costanti: 0 o 1
Variabili: letterali che possono 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) e (A.B)

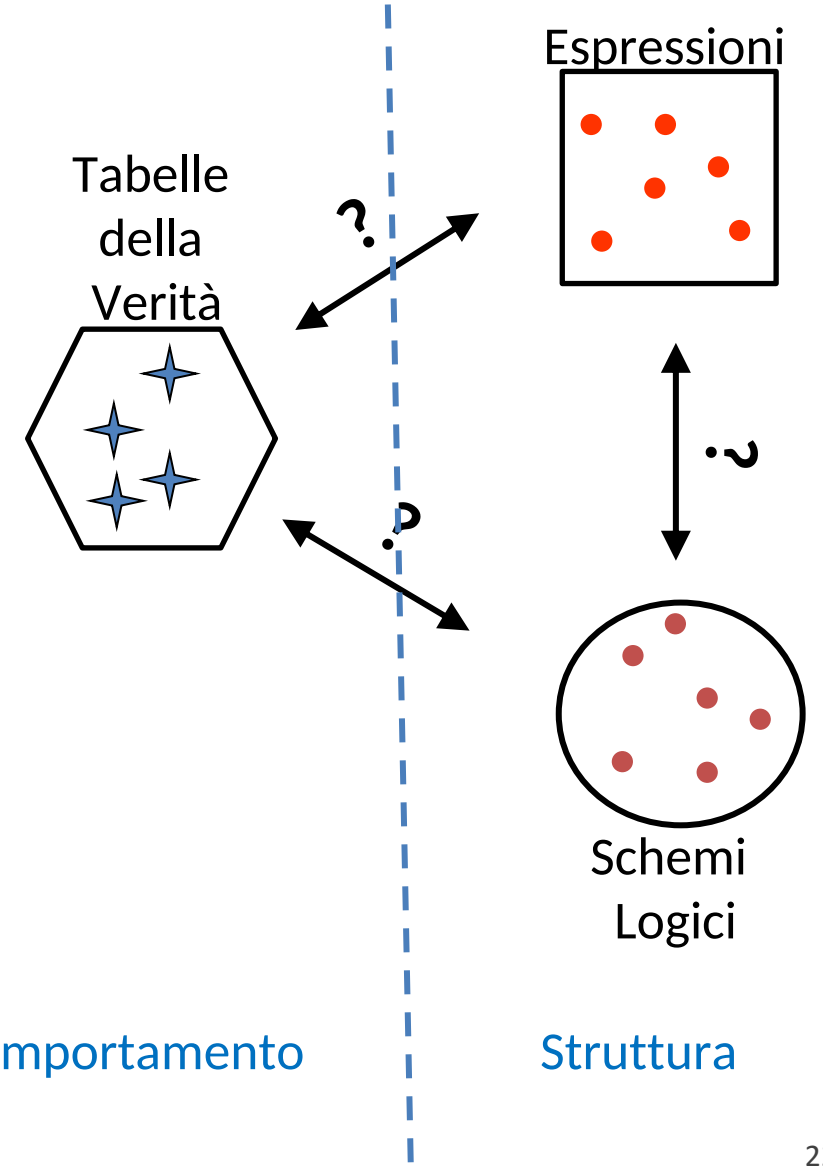
Esempi: a+(b.c) a + bc 0

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

N.B. - L'operazione di negazione è prioritaria rispetto alle altre e quella del prodotto è prioritaria rispetto alla somma. Non è quindi obbligatorio racchiuderla tra parentesi. La notazione AB indica A·B

Tabelle della verità, espressioni e schemi

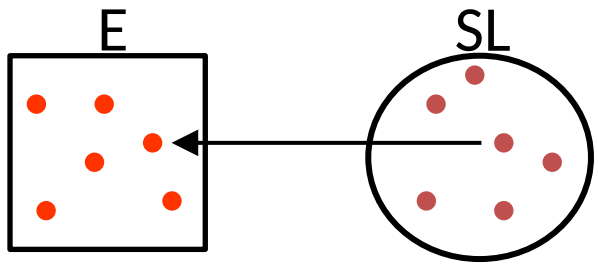
- Qual è la relazione tra tabelle della verità, che descrivono il comportamento, e espressioni e schemi logici che descrivono la struttura di una rete logica?
- Quante espressioni sono possibili data una tabella della verità? E quanti schemi logici?
- A quante tabelle corrisponde un'espressione? A quante tabelle corrisponde uno schema logico?

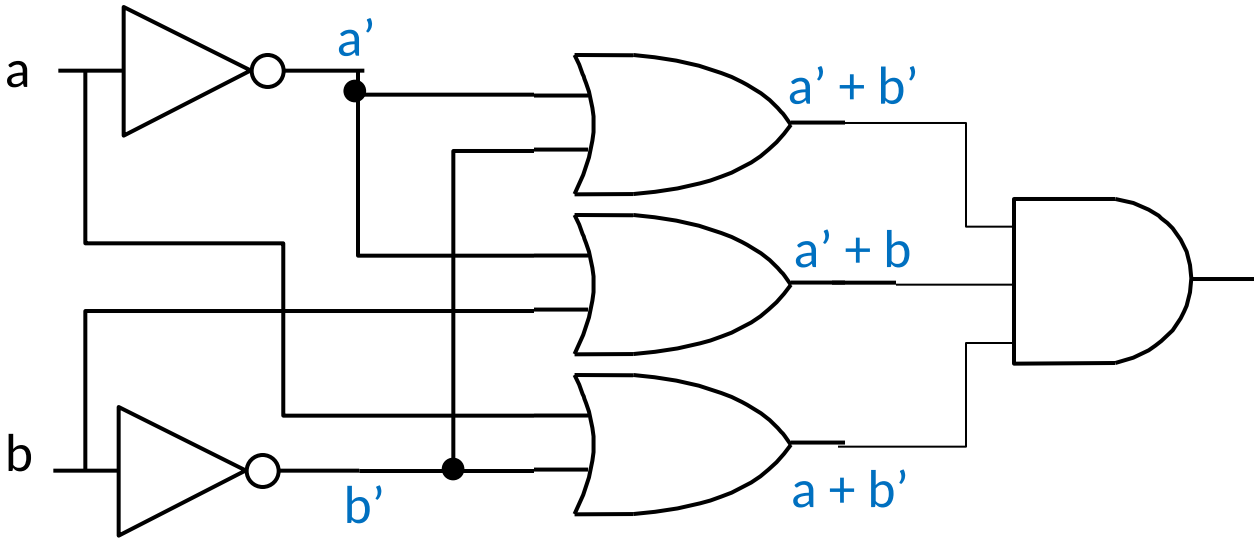


Da Schemi logici a Espressioni

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

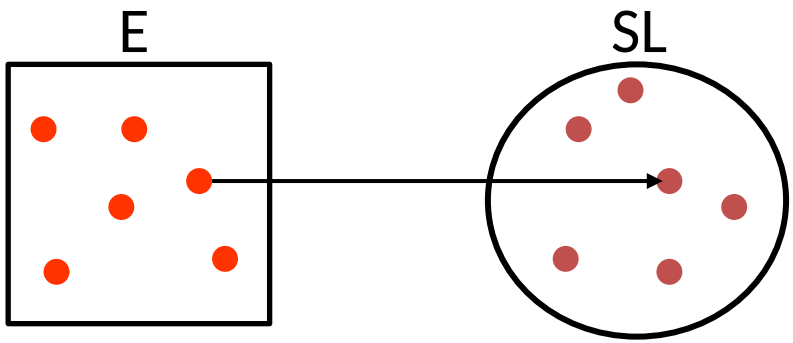
Relazione I: Ogni struttura formata da gate connessi in serie e/o in parallelo è descritta da una sola espressione.





z = (a'+b').(a'+b).(a+b')

Da Espressioni a Schemi logici

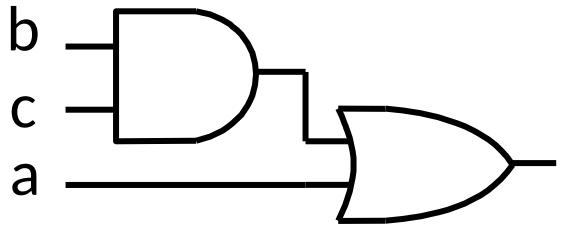


Relazione II: Ogni espressione descrive una sola struttura formata da gate connessi in serie e/o in parallelo.

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.

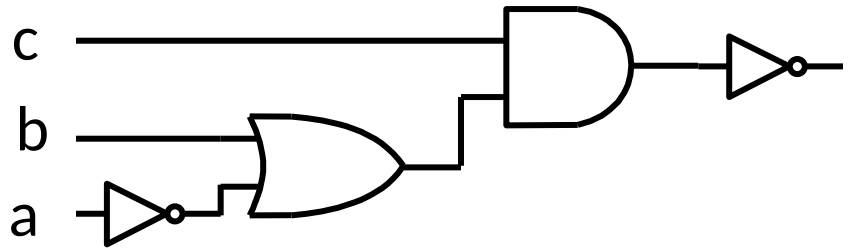
Esempio:

a+(b.c)



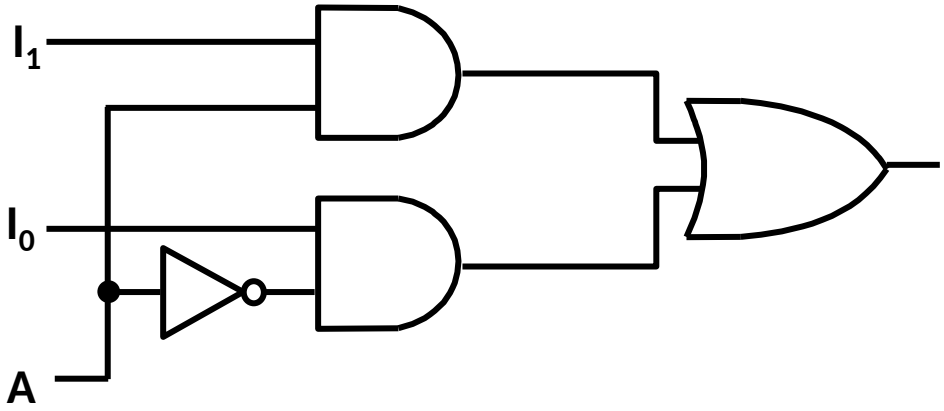
Esempi

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



Selettore a 2 vie (*Multiplexer*)

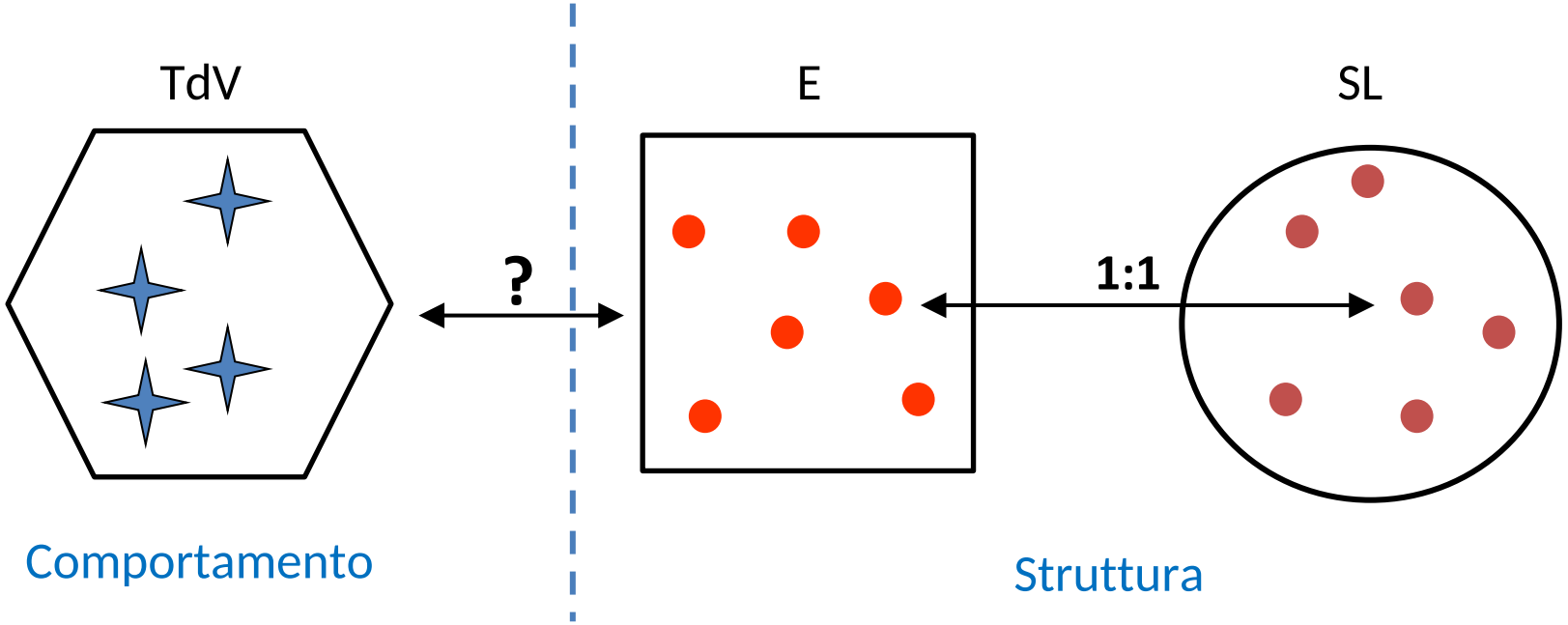
$A' \cdot I_0 + A \cdot I_1$



N.B. - Lo schema logico di una espressione corrispondente ad una rete combinatoria 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").

Espressioni = Schemi Logici

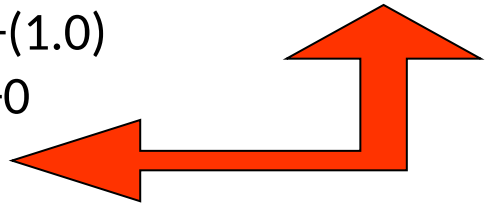
- Esiste quindi una **relazione biunivoca** tra schemi e espressioni
- Per esprimere la struttura con cui si realizza una funzione si può quindi utilizzare indifferentemente l'una o l'altra.
- Quando possibile è preferibile privilegiare l'espressione in quanto più compatta. L'espressione contiene già tutta l'informazione, non è necessario disegnare anche lo schema
- Che relazione c'è tra espressioni (e quindi schemi logici) e funzioni?



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 ha nella n-upla
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, che corrispondono alle n configurazioni binarie dei suoi ingressi.

Quindi, valutando una espressione di n variabili per tutte le possibili 2^n configurazioni, ottengo la **tabella della verità**, ovvero il comportamento della rete combinatoria che ha quella struttura

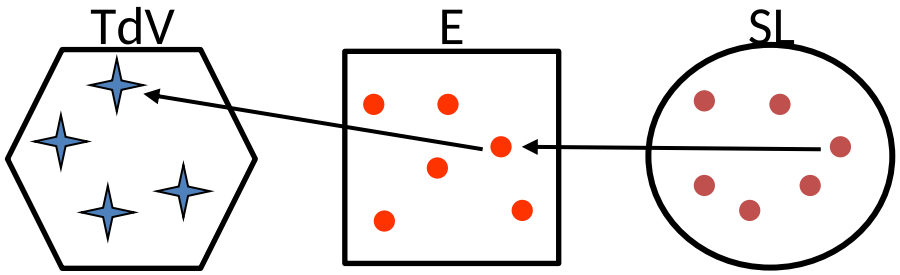
Da Espressioni a TdV

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

Relazione III: Ogni espressione descrive una e una sola tabella della verità **completamente specificata**.



Il procedimento di ANALISI ha quindi un risultato univoco: partendo da un dato schema logico o da un'espressione, si può determinare in modo univoco il comportamento (la tabella della verità) che realizzano

Analisi: da struttura a comportamento

Comportamento
(cosa fa la rete)

sintesi

analisi

Descrizione in linguaggio naturale
«la rete ha uscita z=1 quando x è uguale a y e diverso da w»

Tabella della verità

w	x	y	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
…	…	…	…

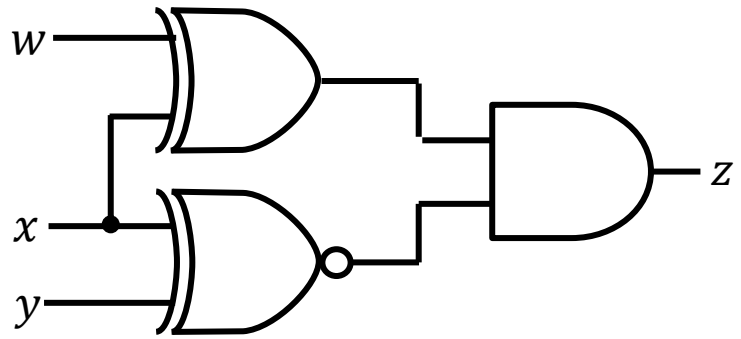
Struttura
(come è realizzata la rete)

Espressione

$$z = (x \equiv y)(x \oplus w)$$

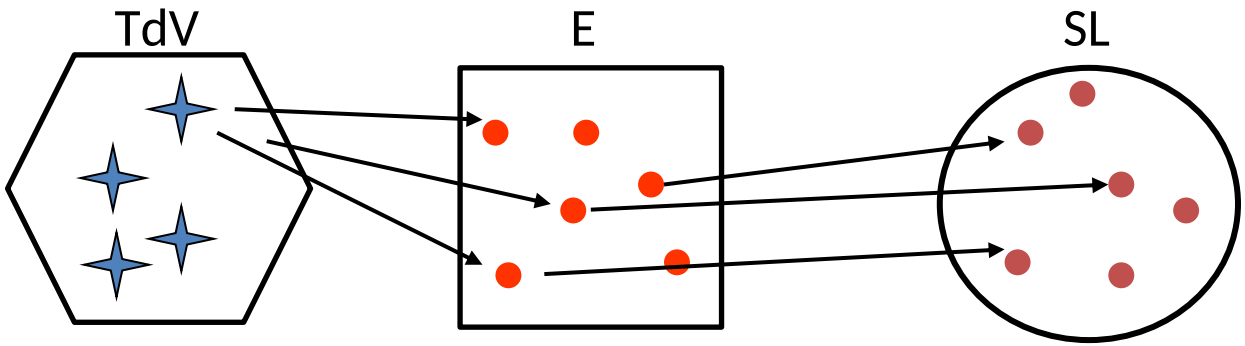
o, in modo equivalente,

Schema logico



Da TdV a Espressioni

Relazione IV: Una funzione può essere descritta da una molteplicità di espressioni



- **Problema della SINTESI**: partendo da un comportamento dato come tabella della verità, quale scegliere tra i vari schemi logici (o espressioni) che lo realizzano?
- Uno dei metodi più semplici per passare da una tabella della verità a una espressione (tra le possibili) riguarda l'utilizzo delle cosiddette «**espressioni canoniche**»

TdV e Espressioni (canoniche)

Espressione canonica SP *(Somma di Prodotti) o I^a forma canonica*

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

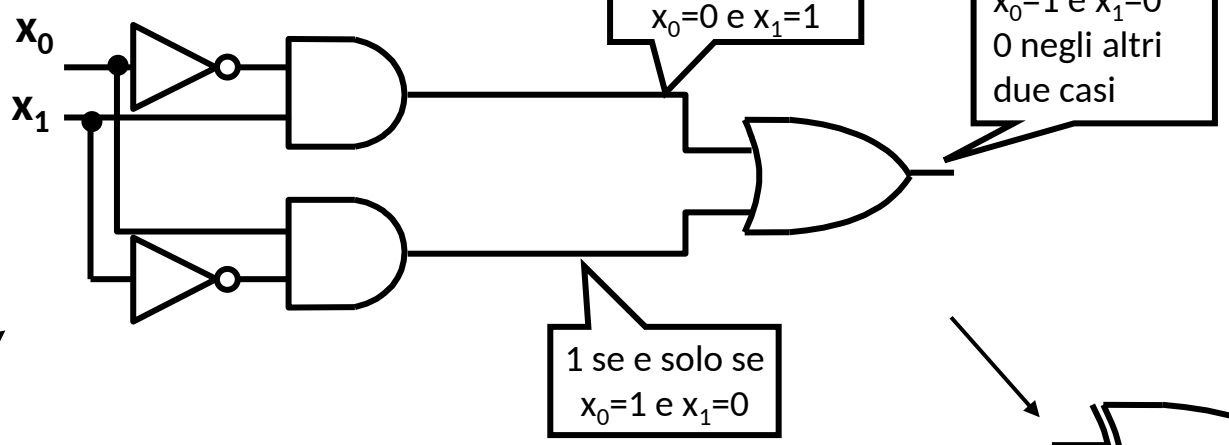
Espressione canonica PS *(Prodotto di Somme) o II^a forma canonica*

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

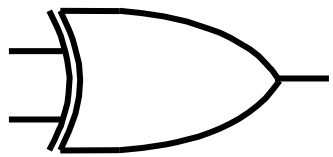
N.B.: questo significa che **ogni** rete combinatoria può in principio essere realizzata **con due soli livelli di gate** (non si considerano i NOT). Ovviamente sono possibili altre realizzazioni, preferibili per altre caratteristiche.

Sintesi canonica del EX-OR

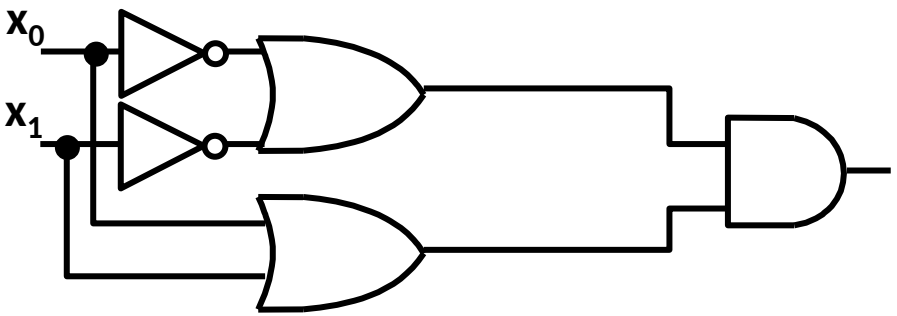
I^a forma canonica (SP):
$F(x_0, x_1)= x_0' x_1 + x_0 x_1'$



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



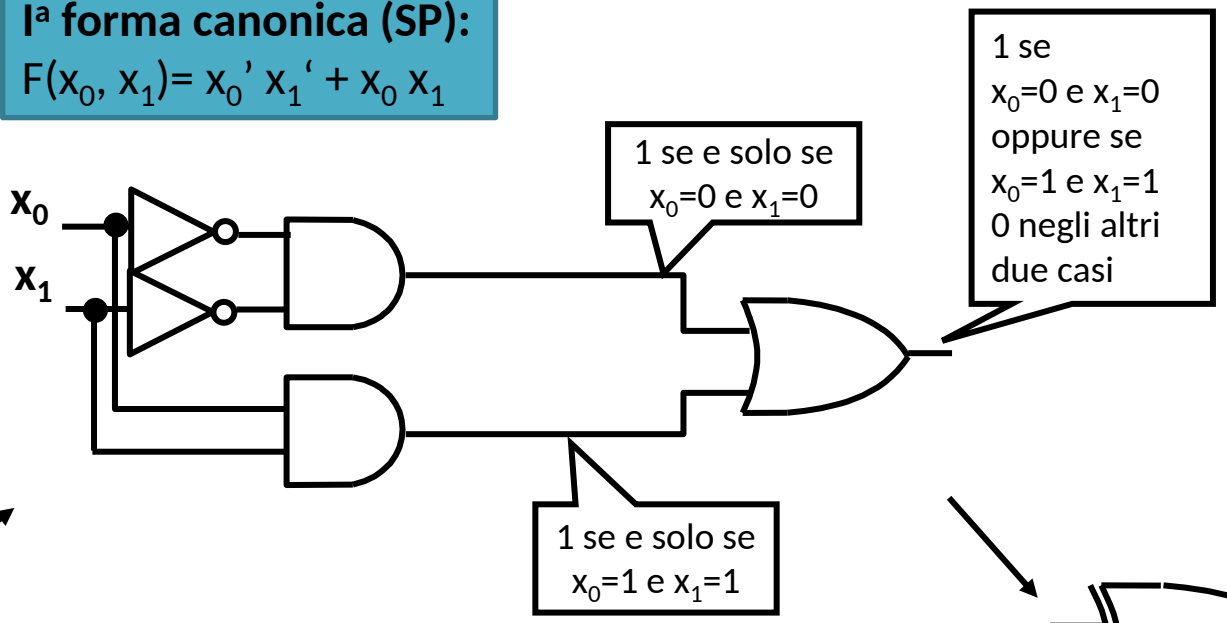
II^a forma canonica (PS):
$F(x_0, x_1)= (x_0+x_1)(x_0'+ x_1')$



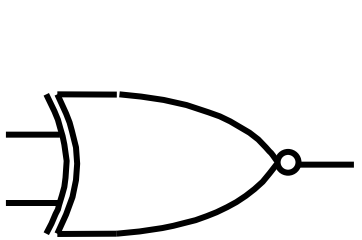
Realizzare e simulare con Digital

Sintesi canonica dell'Equivalence

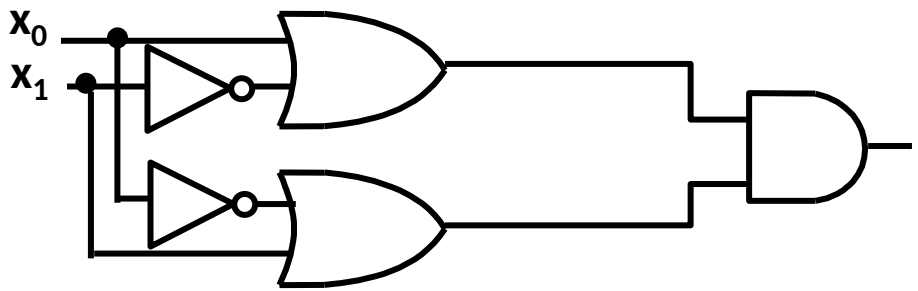
I^a forma canonica (SP):
$F(x_0, x_1)= x_0' x_1' + x_0 x_1$



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



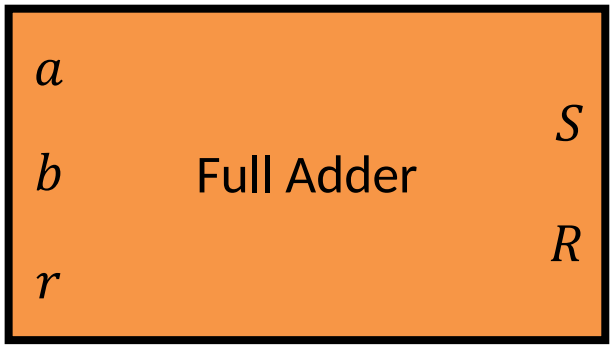
II^a forma canonica (PS):
$F(x_0, x_1)= (x_0+x_1')(x_0'+ x_1)$



Esempio: full adder

- Definire la struttura di una rete logica con 3 ingressi a, b , r e 2 uscite S e R che presenta
  - uscita $S = 1$ quando il numero di «1» sui suoi ingressi è dispari
  - uscita $R = 1$ quando in ingresso ci sono due o più «1».
- Questa rete è combinatoria perché l'uscita dipende solo dagli ingressi attuali
- Questa rete è un **«Full Adder»**: vedremo più avanti che è un componente fondamentale per realizzare operazioni aritmetiche tra numeri binari

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

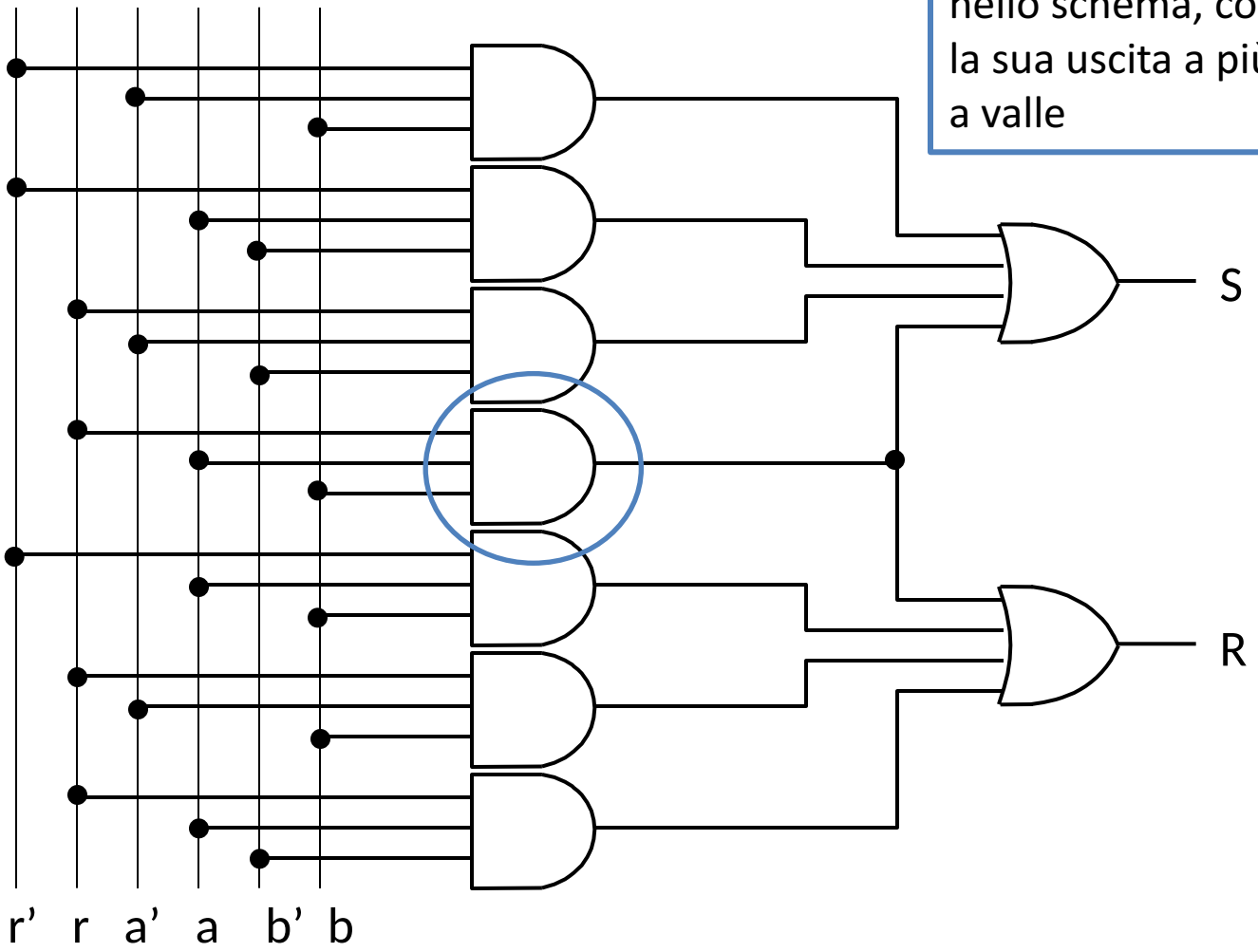
R = a'br + ab'r + abr' + abr

Sintesi canonica (1ª forma) del Full Adder

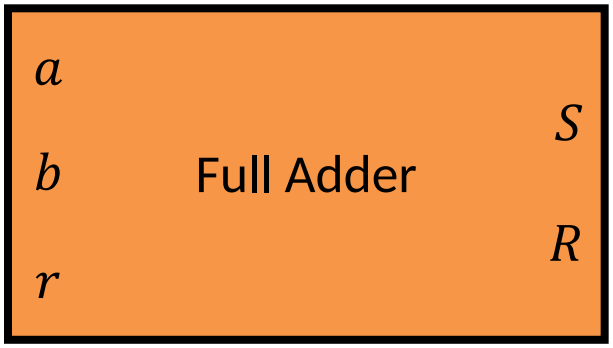
$S = a' b' r + a' b r' + a b' r' + a b r$

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

Se compare lo stesso gate in più espressioni, posso riportarlo una volta sola nello schema, collegando la sua uscita a più ingressi a valle



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))

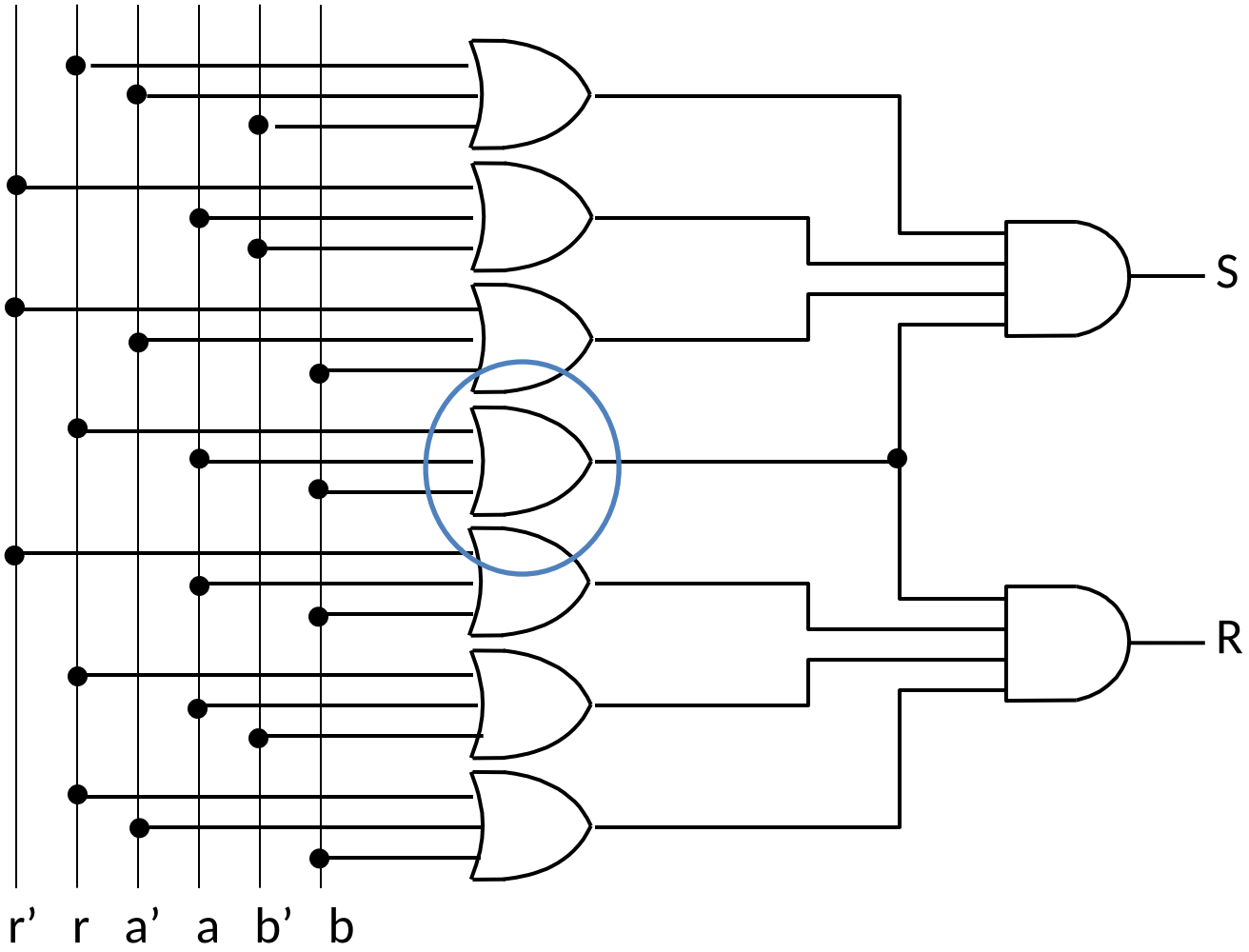
S = (a+b+r) (a+b'+r') (a'+b+r') (a'+b'+r)

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

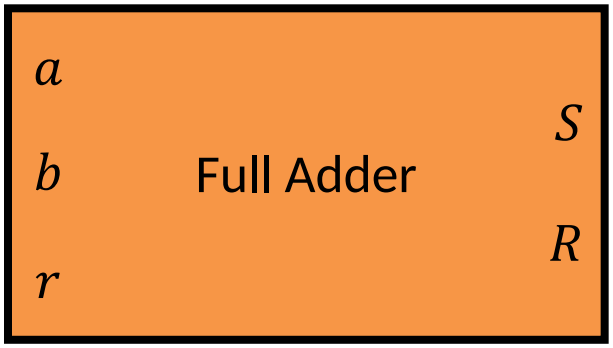
Sintesi canonica (2ª forma) del Full Adder

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)



Espressioni canoniche: notazione simbolica



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

$S(a,b,r) = \Sigma_3\, m(1,2,4,7)$

$= \Pi_3\, M(0,3,5,6)$

$R(a,b,r) = \Sigma_3\, m(3,5,6,7)$

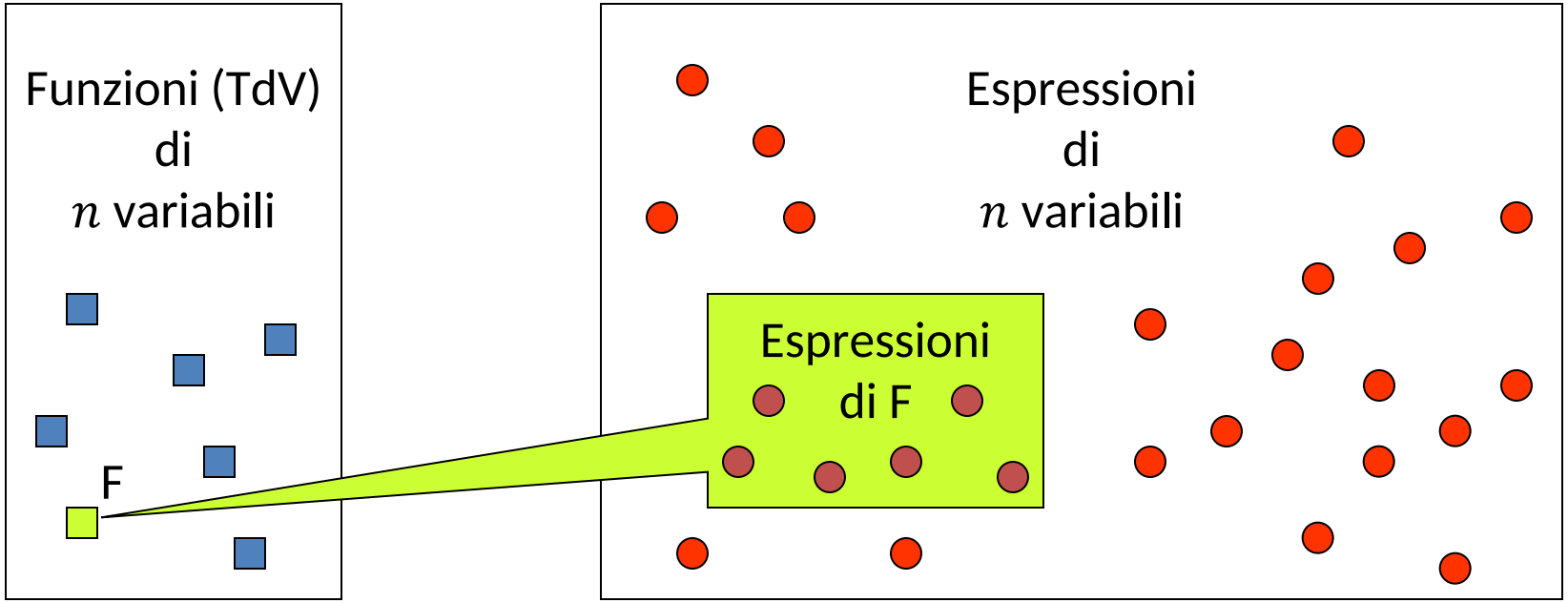
$= \Pi_3\, M(0,1,2,4)$

- 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
- Mintermini e maxtermini sono complementari: una configurazione è un mintermine o un maxtermine

- 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 (o TdV).



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

Equivalenza tra espressioni

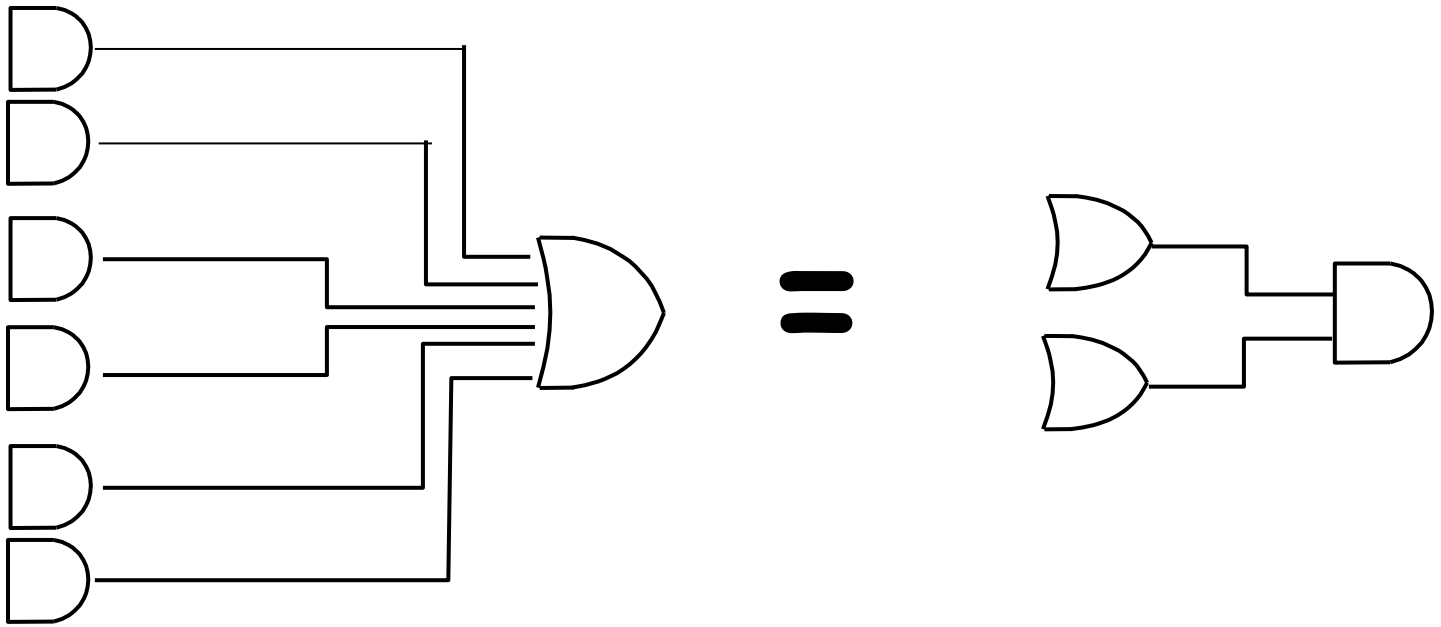
Espressioni equivalenti possono corrispondere a schemi logici di **complessità differente**

Esempio: funzione U di 3 variabili a, b, c con 6 mintermini e 2 maxtermini:

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

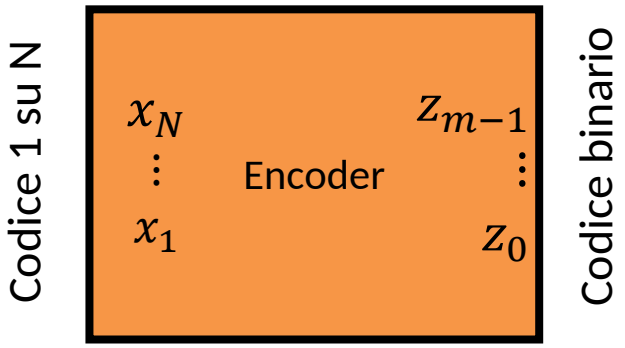
Le sue espressioni canoniche richiedono

- 6 AND e 1 OR nella forma SP
- 2 OR e 1 AND nella forma PS



Espressioni di funzioni incomplete

- Anche espressioni che forniscono eguale valutazione **limitatamente al dominio di una funzione incompleta** sono dette **equivalenti**.
- Esempio: **l'encoder** è una rete che realizza la trascodifica da codice 1 su N a codice binario.
- A seconda del valore assegnato alle configurazioni non utilizzate dalla funzione che realizza un encoder a 3 ingressi, ottengo una **famiglia di espressioni** diverse tra loro equivalenti



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, quindi non è definito come risponde la rete

Espressioni di funzioni incomplete

- Riempiendo le configurazioni non utilizzate con degli «uni» anziché degli «zeri» posso ottenere un'espressione equivalente, ma più semplice, delle due uscite. **Come individuare la scelta ottima?**

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
0	1	1	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

$$z_1 = x_3 x_2' x_1' + x_3' x_2 x_1'$$
$$z_0 = x_3 x_2' x_1' + x_3' x_2' x_1$$

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
0	1	1	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

$$z_1 = x_3 + x_2$$
$$z_0 = x_3 + x_1$$

Sintesi di reti combinatorie

- Un possibile approccio per realizzare la sintesi di una funzione data è il seguente:
  - Scegliere una espressione tra le molteplici corrispondenti a una data funzione (es. le espressioni canoniche)
  - 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:
(tutte verificabili confrontando le tabelle della verità dei due lati delle uguaglianze)

E1) *commutativa*

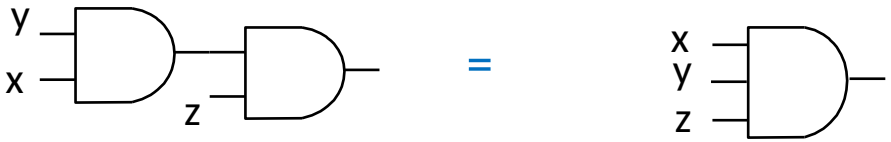
$x + y = y + x$

$x \cdot y = y \cdot x$

E2) *associativa*

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

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



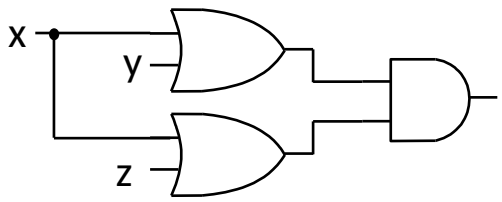
(utile per ridurre il fan-in)

Equivalenze notevoli

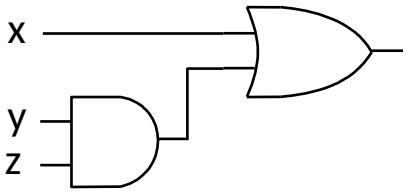
E3) *distributiva*

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

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



=



N.B.: Questa equivalenza, non valida in algebra «tradizionale», è vera in algebra binaria. In generale, in algebra binaria ogni proprietà è **duale**.

Verifica con TdV

(riduzione del numero di gate utilizzati)

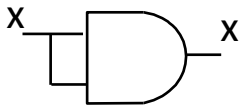
x	y	z	a=x+y	b=x+z	a·b	c=y·z	x+c
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Equivalenze notevoli

E4) *idempotenza*

$x + x = x$

$x \cdot x = x$

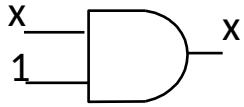


x	x·x
0	0
1	1

E5) *identità*

$x + 0 = x$

$x \cdot 1 = x$

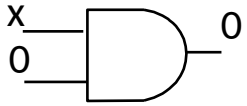


x	x·1
0	0
1	1

E6) *limite*

$x + 1 = 1$

$x \cdot 0 = 0$

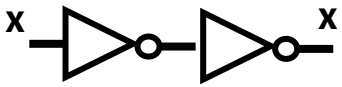


x	x·0
0	0
1	0

Equivalenze notevoli

<u>Proprietà della complementazione</u>:

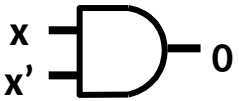
E7) *involuzione* $(x')' = x$



(amplificazione di segnale)

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

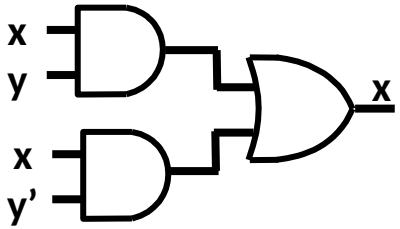
$x \cdot x' = 0$



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)

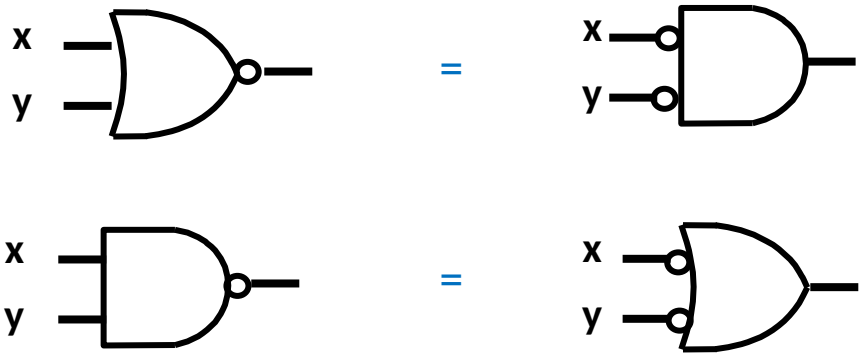
$= x$ (E5 – identità)

(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

- utili 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:

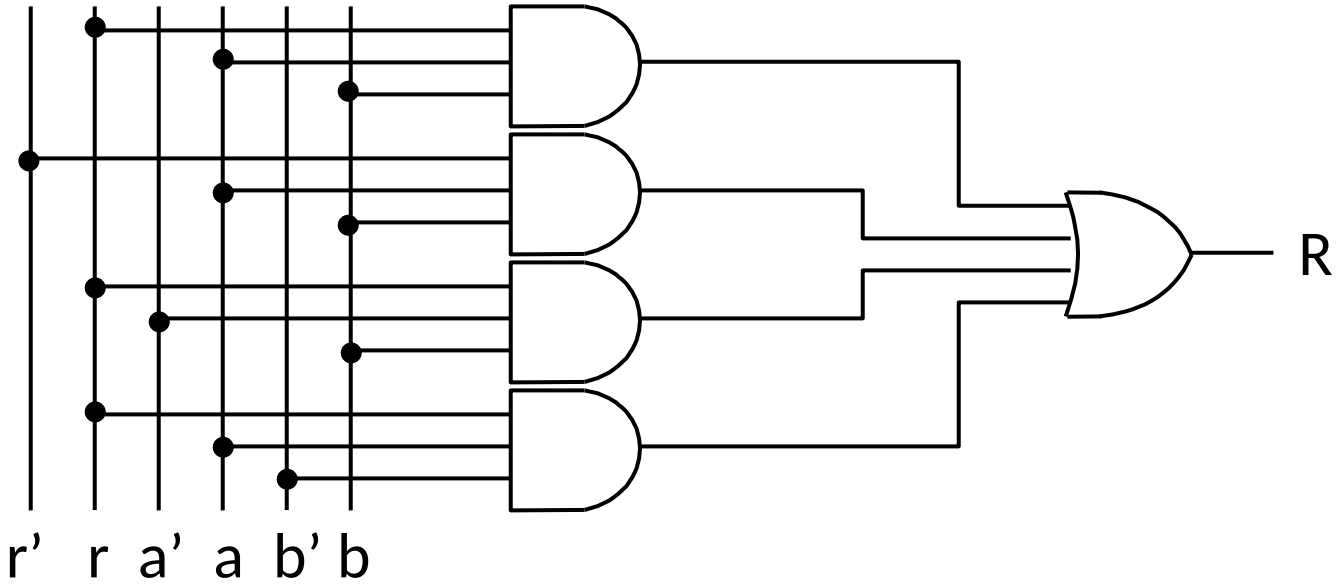
$xy + x'z + yz$	$= xy + x'z + yz\,(x + x')$	Limitazione e Identità
	$= xy + x'z + yzx + yzx'$	Distributiva
	$= xy\,(1+z) + x'z\,(1+y)$	Distributiva
	$= xy\,(1) + x'z\,(1)$	Limite
	$= xy + x'z$	Identità

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

Manipolazione algebrica di espressioni …

- Lo schema logico 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 e 3 NOT per avere gli ingressi in forma vera e negata.

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



Manipolazione algebrica di espressioni …

- È possibile ottenere una espressione più semplice? Una possibile manipolazione.

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

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

$= a' b r + a b' r + a b 1$ — Limitazione (E_8)

$= a' b r + a b' r + a b$ — Identità (E_5)

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

Formulazione SP originale..

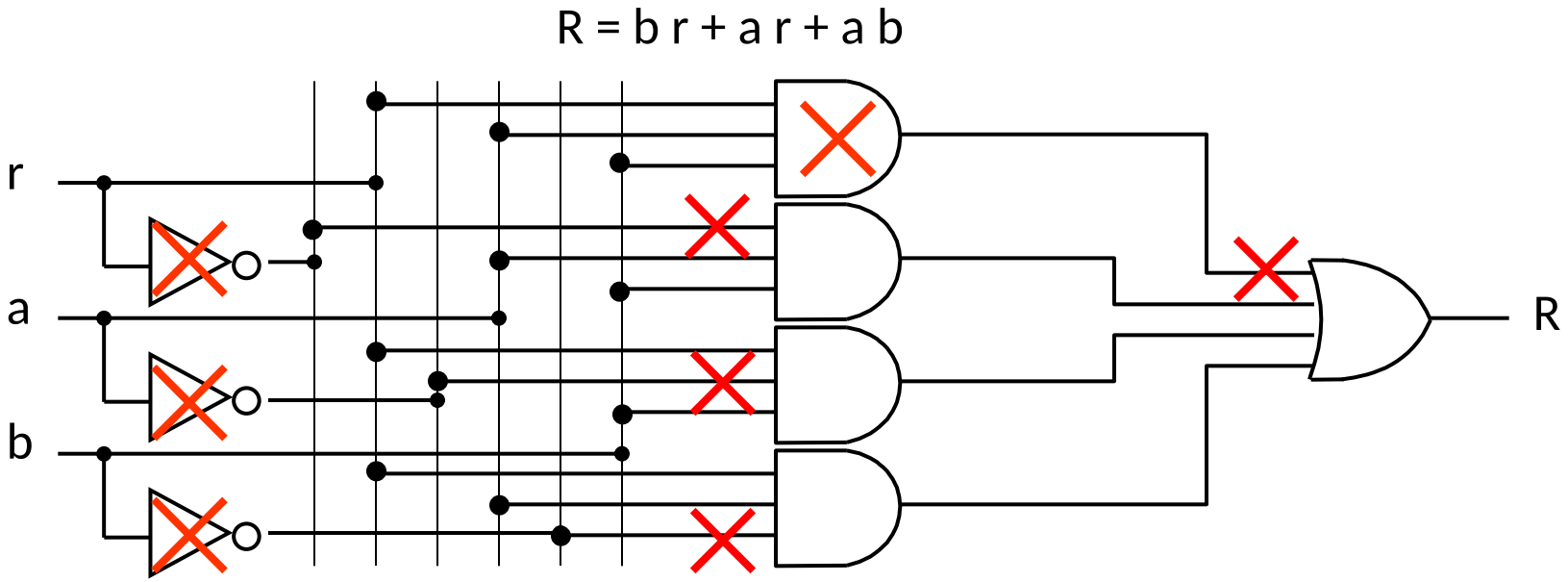
$R = a' b r + a b' r + a b r' + a b r + a b r + a b r$ — Idempot. (E_4)

$= b r (a' + a) + a r (b' + b) + a b (r' + r)$ — 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

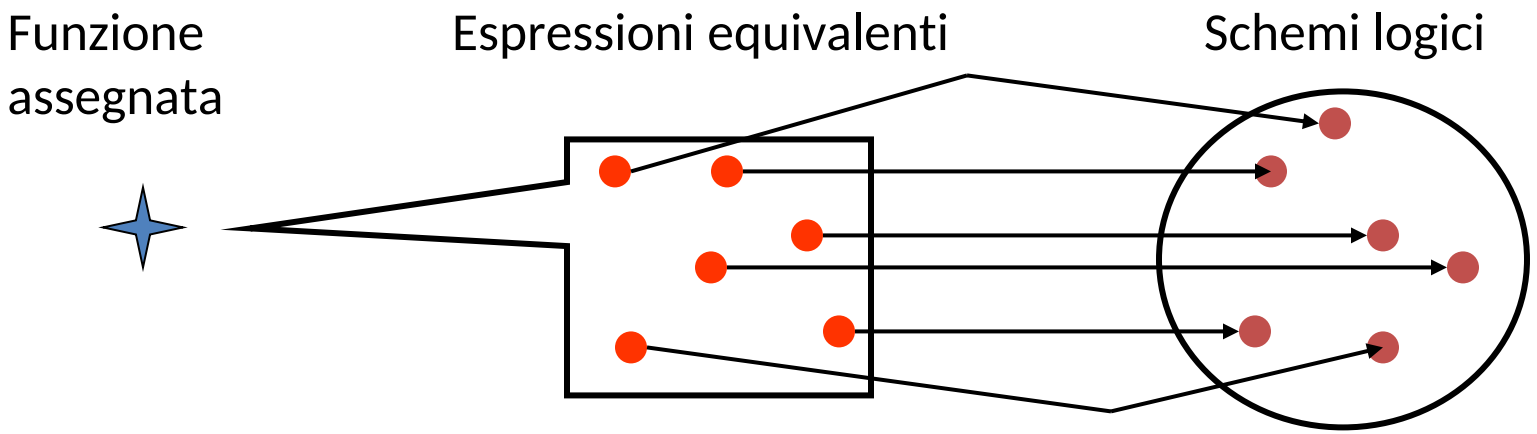


1 OR a 4 ingressi
4 AND a 3 ingressi
3 NOT

1 OR a 3 ingressi
3 AND a 2 ingressi

- Come abbiamo visto il procedimento di manipolazione algebrica verso la rete *«di costo minimo»* può non essere affatto ovvio
- Il processo di sintesi volto all'ottenimento della rete di costo minimo viene realizzato mediante opportune metodologie (es. le mappe di Karnaugh, si vedranno più avanti)

Il problema della sintesi



SINTESI: individuazione **dell'espressione** che fornisce lo schema **"migliore"** per la realizzazione della funzione assegnata. «Migliore» può essere definito con criteri anche opposti tra loro, quali

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

Progetto logico di circuiti combinatori