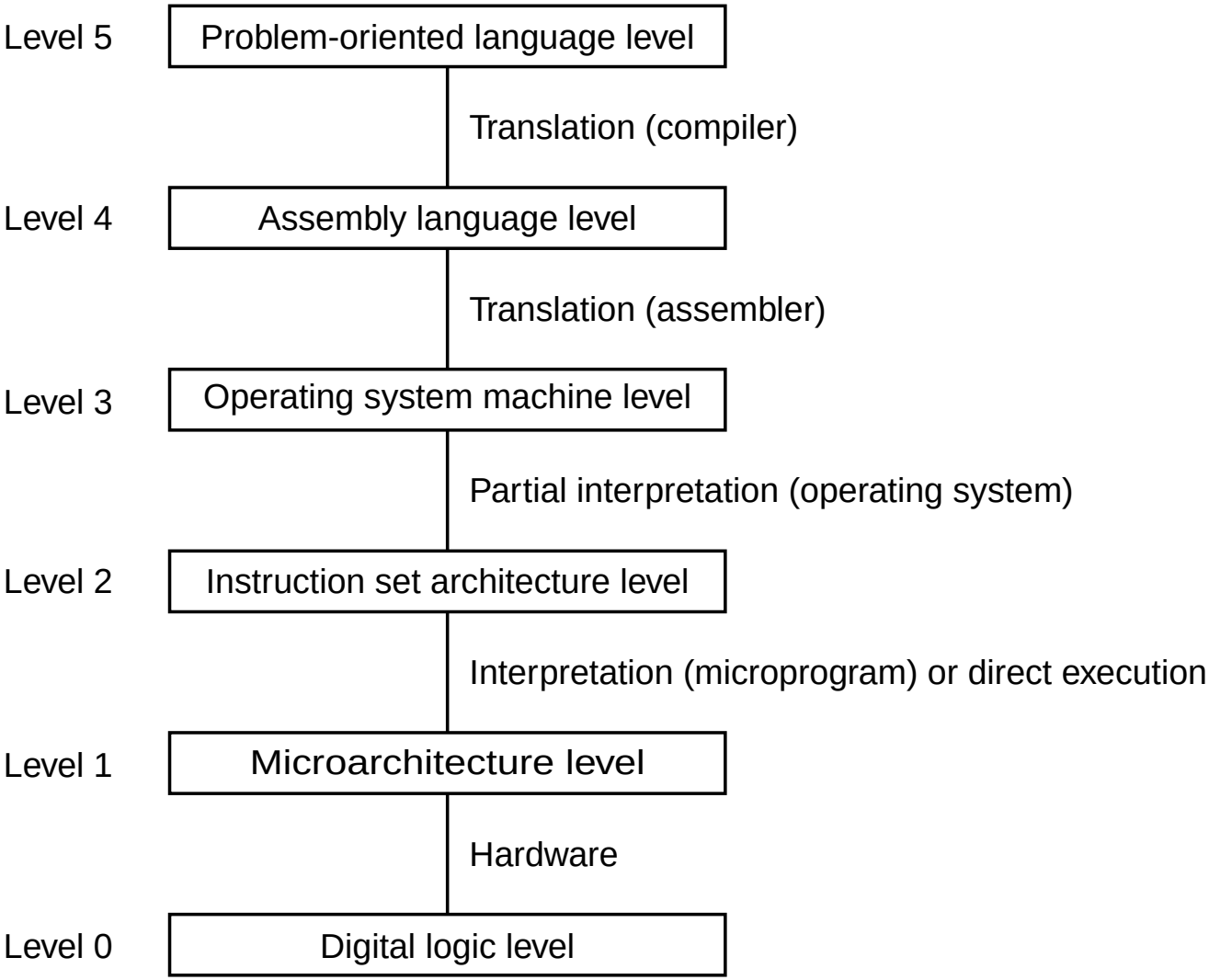
Livello 0: logica binaria



Segnali: dominio analogico e digitale

Nei calcolatori le informazioni sono rappresentate e trasmesse da segnali elettrici.
Due metodi per rappresentare temporalmente l'informazione tramite segnali.

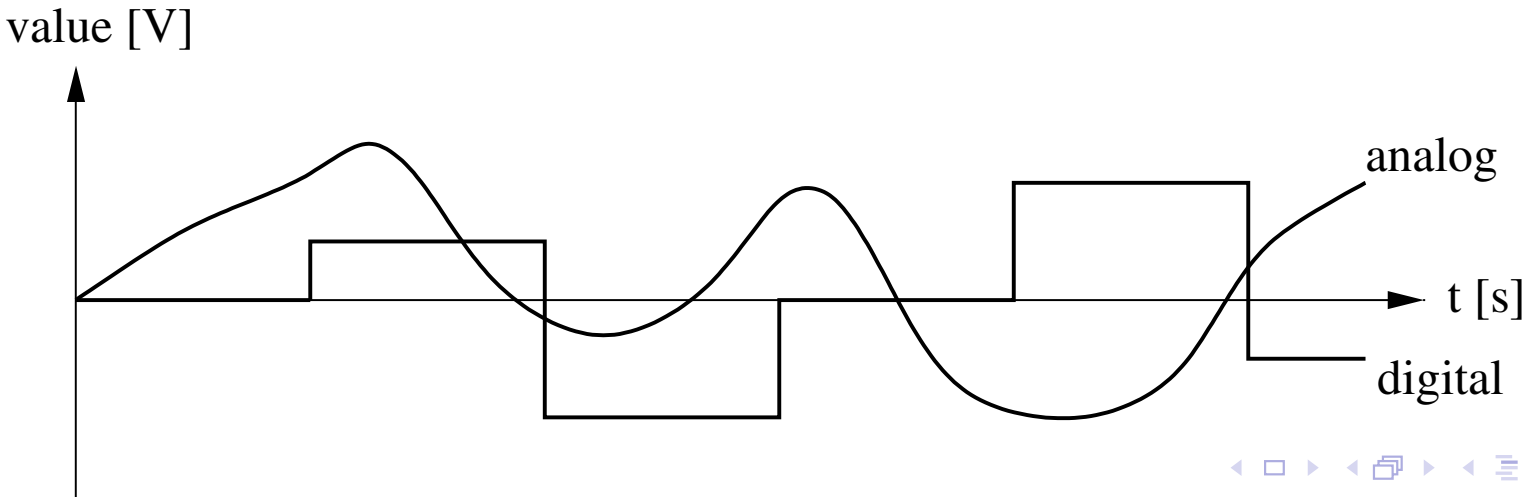
- segnale analogico: infiniti valori (es.: telefono)

Segnali: dominio analogico e digitale

Nei calcolatori le informazioni sono rappresentate e trasmesse da segnali elettrici.
Due metodi per rappresentare temporalmente l'informazione tramite segnali.

- segnale analogico: infiniti valori (es.: telefono)
- segnale digitale: *n* valori (es.: telegrafo).

In entrambi i casi, l'informazione è associata al livello (positivo o negativo!) di tensione elettrica: Volt (V).



Segnali analogico vs. digitale

Il segnale analogico può teoricamente contenere infinite informazioni in un tempo infinitesimo. Questa possibilità è resa in pratica impossibile a causa della presenza del rumore di segnale, che altera i valori di tensione.

Viceversa, l'informazione nel segnale digitale occupa un unico valore lungo ogni slot temporale. Quindi, il rumore può essere reso ininfluente associando informazioni adiacenti a valori di tensione sufficientemente distanti e/o slot temporali sufficientemente estesi.

Dunque: più rumore c'è, meno informazione riesco a trasmettere nell'unità di tempo.

Rappresentazione di bit da segnali

Il segnale digitale inoltre si sposa meglio con l'informazione nel calcolatore, che non è mai infinita.
Es.: 1 byte = 2^8 valori $\rightarrow$ ogni slot temporale trasporta 1 byte se utilizzo 256 livelli di tensione

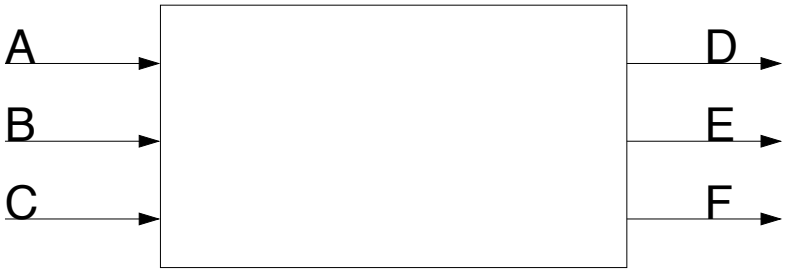
Spesso (ma non sempre!) i segnali digitali nel computer sono binari, cioè rappresentano due soli valori a ogni slot. Es.: $A = \{0, 1\}$ a ogni slot.
Logica positiva:

- "alta" tensione (5 V o -5 V): valore 1
- "bassa" tensione: ($\sim$ 0 Volt): valore 0.

Logica negativa: i valori 0 e 1 sono complementati.
Altre possibili "alte" tensioni: ± 2.2 V, ± 1.25 V.

Circuiti logici

Mappano istantaneamente (cioè: a ogni slot)
k segnali binari in ingresso (input, es. A, B, C) in
m segnali binari in uscita (output, es. D, E, F).



Es.: $f(0, 1, 1) = (1, 0, 1)$.

Possono essere

- senza memoria: $(D, E, F) = f(A, B, C)$
- con memoria: $(D, E, F) = f(A, B, C, \text{memoria})$.

Approfondiremo il comportamento in assenza e in presenza di memoria interna al circuito logico.

Circuiti combinatori e sequenziali

La presenza o meno di memoria individua due classi di circuiti logici:

- combinatori: l'uscita $\boldsymbol{y}_n$ allo slot n-esimo dipende solo dall'ingresso $\boldsymbol{x}_n$ allo stesso slot: $\boldsymbol{y}_n = f(\boldsymbol{x}_n)$

Circuiti combinatori e sequenziali

La presenza o meno di memoria individua due classi di circuiti logici:

- combinatori: l'uscita $\boldsymbol{y}_n$ allo slot n-esimo dipende solo dall'ingresso $\boldsymbol{x}_n$ allo stesso slot: $\boldsymbol{y}_n = f(\boldsymbol{x}_n)$
- sequenziali: l'ingresso $\boldsymbol{x}_n$ e lo stato della memoria $\boldsymbol{s}_n$ allo slot n-esimo determinano, oltre all'uscita $\boldsymbol{y}_n$, anche lo stato allo slot successivo:
$$\begin{cases} \boldsymbol{y}_n = g(\boldsymbol{x}_n, \boldsymbol{s}_n) \\ \boldsymbol{s}_{n+1} = q(\boldsymbol{x}_n, \boldsymbol{s}_n) = \ldots = h(\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \ldots) \end{cases}$$
da cui $\boldsymbol{y}_n = f(\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \ldots)$.
Ciò rende l'uscita dai circuiti sequenziali dipendente dall'intero segnale d'ingresso.

Porte logiche

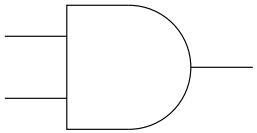
Una porta logica realizza un circuito combinatorio a una uscita, avente una semplice interpretazione logica in cui scegliamo $0 \leftrightarrow$ FALSO, $1 \leftrightarrow$ VERO.

Le porte logiche di più immediata interpretazione realizzano operativamente le particelle linguistiche 'E' (AND), 'O' (OR), 'NON' (NOT):

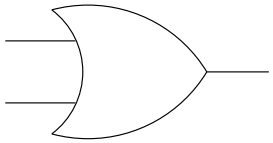
- piove e fa freddo
- piove o fa freddo
- non piove.

Gli ingressi possono essere più di due: piove e fa freddo e scende la sera.

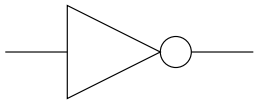
Porte AND, OR, NOT



AND



OR



NOT

- AND ha uscita 1 se tutti gli ingressi sono a 1 (e 0 altrimenti): vera se piove E se fa freddo.
- OR ha uscita 1 se almeno uno degli ingressi è a 1 (e 0 altrimenti): vera O se piove O se fa freddo.
- NOT ha uscita 1 se l'ingresso ha valore 0 (e 0 altrimenti): vera se NON piove.

Realizzazione di porte logiche

Le porte logiche sono dei piccoli circuiti elettronici di cui forniamo un'idea intuitiva attingendo dall'idraulica, le cui componenti e leggi sono equivalenti.

Realizzazione di porte logiche

Le porte logiche sono dei piccoli circuiti elettronici di cui forniamo un'idea intuitiva attingendo dall'idraulica, le cui componenti e leggi sono equivalenti.

- filo elettrico $\leftrightarrow$ tubo, condotta

Realizzazione di porte logiche

Le porte logiche sono dei piccoli circuiti elettronici di cui forniamo un'idea intuitiva attingendo dall'idraulica, le cui componenti e leggi sono equivalenti.

- filo elettrico $\leftrightarrow$ tubo, condotta
- tensione elettrica $\leftrightarrow$ pressione sul tubo

Realizzazione di porte logiche

Le porte logiche sono dei piccoli circuiti elettronici di cui forniamo un'idea intuitiva attingendo dall'idraulica, le cui componenti e leggi sono equivalenti.

- filo elettrico $\leftrightarrow$ tubo, condotta
- tensione elettrica $\leftrightarrow$ pressione sul tubo
- corrente elettrica $\leftrightarrow$ flusso di liquido nel tubo

Realizzazione di porte logiche

Le porte logiche sono dei piccoli circuiti elettronici di cui forniamo un'idea intuitiva attingendo dall'idraulica, le cui componenti e leggi sono equivalenti.

- filo elettrico $\leftrightarrow$ tubo, condotta
- tensione elettrica $\leftrightarrow$ pressione sul tubo
- corrente elettrica $\leftrightarrow$ flusso di liquido nel tubo
- resistenza elettrica $\leftrightarrow$ strozzatura nel tubo

Realizzazione di porte logiche

Le porte logiche sono dei piccoli circuiti elettronici di cui forniamo un'idea intuitiva attingendo dall'idraulica, le cui componenti e leggi sono equivalenti.

- filo elettrico $\leftrightarrow$ tubo, condotta
- tensione elettrica $\leftrightarrow$ pressione sul tubo
- corrente elettrica $\leftrightarrow$ flusso di liquido nel tubo
- resistenza elettrica $\leftrightarrow$ strozzatura nel tubo
- batteria/alimentatore $\leftrightarrow$ pompa

Realizzazione di porte logiche

Le porte logiche sono dei piccoli circuiti elettronici di cui forniamo un'idea intuitiva attingendo dall'idraulica, le cui componenti e leggi sono equivalenti.

- filo elettrico $\leftrightarrow$ tubo, condotta
- tensione elettrica $\leftrightarrow$ pressione sul tubo
- corrente elettrica $\leftrightarrow$ flusso di liquido nel tubo
- resistenza elettrica $\leftrightarrow$ strozzatura nel tubo
- batteria/alimentatore $\leftrightarrow$ pompa
- interruttore $\leftrightarrow$ saracinesca

Realizzazione di porte logiche

Le porte logiche sono dei piccoli circuiti elettronici di cui forniamo un'idea intuitiva attingendo dall'idraulica, le cui componenti e leggi sono equivalenti.

- filo elettrico $\leftrightarrow$ tubo, condotta
- tensione elettrica $\leftrightarrow$ pressione sul tubo
- corrente elettrica $\leftrightarrow$ flusso di liquido nel tubo
- resistenza elettrica $\leftrightarrow$ strozzatura nel tubo
- batteria/alimentatore $\leftrightarrow$ pompa
- interruttore $\leftrightarrow$ saracinesca
- condensatore $\leftrightarrow$ serbatoio

Realizzazione di porte logiche

Le porte logiche sono dei piccoli circuiti elettronici di cui forniamo un'idea intuitiva attingendo dall'idraulica, le cui componenti e leggi sono equivalenti.

- filo elettrico $\leftrightarrow$ tubo, condotta
- tensione elettrica $\leftrightarrow$ pressione sul tubo
- corrente elettrica $\leftrightarrow$ flusso di liquido nel tubo
- resistenza elettrica $\leftrightarrow$ strozzatura nel tubo
- batteria/alimentatore $\leftrightarrow$ pompa
- interruttore $\leftrightarrow$ saracinesca
- condensatore $\leftrightarrow$ serbatoio
- transistor $\leftrightarrow$ valvola.

Il transistor nelle porte logiche

Come una valvola idraulica, il transistor (Shockley, Bardeen & Houser, Nobel '56 per la fisica) regola una corrente attraverso una tensione di controllo.

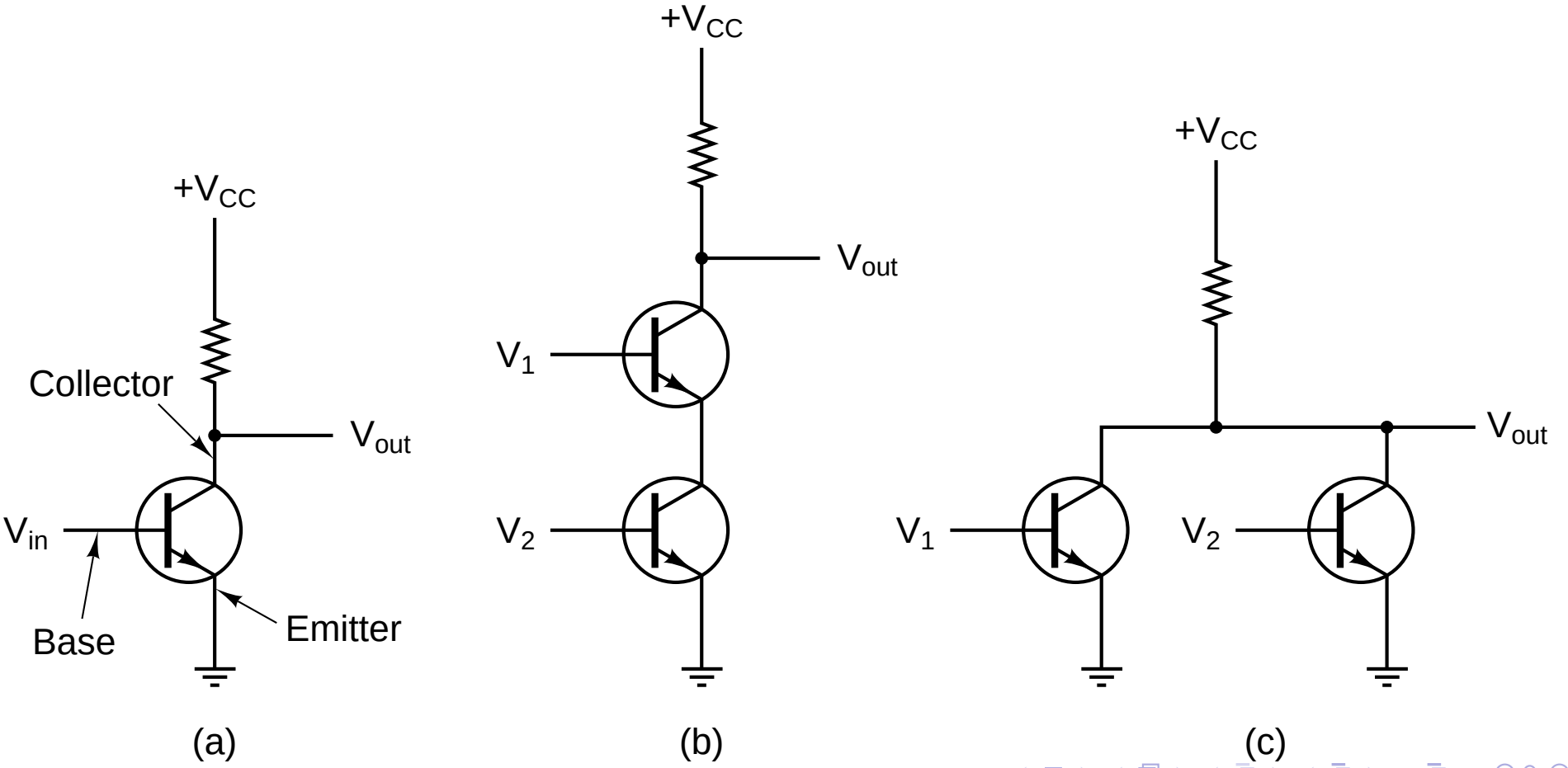


Tabella di verità di un circuito

Descrizione esaustiva del circuito: per ogni combinazione dei valori di ingresso si specifica il valore (determinato oppure no) dell'uscita.

Se ci sono k ingressi, le loro possibili combinazioni sono

Tabella di verità di un circuito

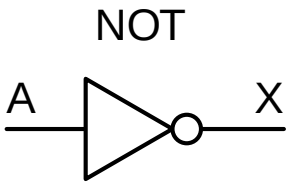
Descrizione esaustiva del circuito: per ogni combinazione dei valori di ingresso si specifica il valore (determinato oppure no) dell'uscita.

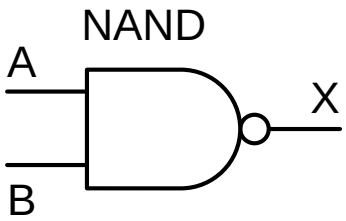
Se ci sono k ingressi, le loro possibili combinazioni sono 2^k. La descrizione completa del circuito può essere data mediante una tabella di verità.

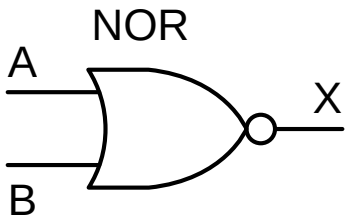
Es.: $(D, E, F) = f(A, B, C)$. 8 combinazioni, 3 uscite

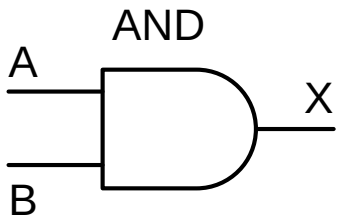
A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	X
0	1	0	1	X	X
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

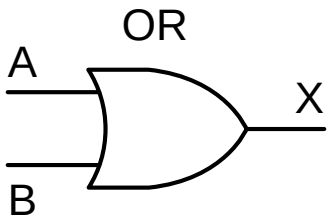
Tabelle di verità di alcune porte logiche











A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

Nota: oltre alla NOT, le realizzazioni coi transistor viste in precedenza erano le porte NAND e NOR
A NAND B equivale a NOT (A AND B)
A NOR B equivale a NOT (A OR B).

Sintesi di circuiti combinatori

Problema: data una tabella di verità di 2^k righe, sintetizzare il circuito combinatorio che la realizza.

Diamo un metodo algoritmico che per ognuna delle m uscite risolve il problema adoperando porte NOT, AND e OR:

1. considero tutte le $p \leq 2^k$ righe della tabella di verità con uscita 1

Sintesi di circuiti combinatori

Problema: data una tabella di verità di 2^k righe, sintetizzare il circuito combinatorio che la realizza.

Diamo un metodo algoritmico che per ognuna delle m uscite risolve il problema adoperando porte NOT, AND e OR:

1. considero tutte le $p \leq 2^k$ righe della tabella di verità con uscita 1
2. per ciascuna di esse prendo una porta AND a k ingressi che restituisca 1; allo scopo applico una porta NOT su ogni ingresso che vale 0

Sintesi di circuiti combinatori

Problema: data una tabella di verità di 2^k righe, sintetizzare il circuito combinatorio che la realizza.

Diamo un metodo algoritmico che per ognuna delle m uscite risolve il problema adoperando porte NOT, AND e OR:

1. considero tutte le $p \leq 2^k$ righe della tabella di verità con uscita 1
2. per ciascuna di esse prendo una porta AND a k ingressi che restituisca 1; allo scopo applico una porta NOT su ogni ingresso che vale 0
3. invio le uscite delle p porte AND a una porta OR a p ingressi: essa è un'uscita del circuito

Sintesi di circuiti combinatori

Problema: data una tabella di verità di 2^k righe, sintetizzare il circuito combinatorio che la realizza.

Diamo un metodo algoritmico che per ognuna delle m uscite risolve il problema adoperando porte NOT, AND e OR:

1. considero tutte le $p \leq 2^k$ righe della tabella di verità con uscita 1
2. per ciascuna di esse prendo una porta AND a k ingressi che restituisca 1; allo scopo applico una porta NOT su ogni ingresso che vale 0
3. invio le uscite delle p porte AND a una porta OR a p ingressi: essa è un'uscita del circuito
4. ripeto i passi 1,2,3 per ognuna delle m uscite.

Sintesi di circuiti: algoritmo duale

Il circuito precedente era un OR di p porte AND. Lo stesso problema può essere risolto dualmente mediante un AND di $\overline{p} = 2^k - p$ porte OR.

1. considero tutte le $\overline{p}$ righe della tabella di verità con uscita 0

Sintesi di circuiti: algoritmo duale

Il circuito precedente era un OR di p porte AND. Lo stesso problema può essere risolto dualmente mediante un AND di $\overline{p} = 2^k - p$ porte OR.

1. considero tutte le $\overline{p}$ righe della tabella di verità con uscita 0
2. per ciascuna di esse prendo una porta OR a k ingressi che restituisca 0; allo scopo applico una porta NOT su ogni ingresso che vale 1

Sintesi di circuiti: algoritmo duale

Il circuito precedente era un OR di p porte AND. Lo stesso problema può essere risolto dualmente mediante un AND di $\overline{p} = 2^k - p$ porte OR.

1. considero tutte le $\overline{p}$ righe della tabella di verità con uscita 0
2. per ciascuna di esse prendo una porta OR a k ingressi che restituisca 0; allo scopo applico una porta NOT su ogni ingresso che vale 1
3. invio le uscite delle $\overline{p}$ porte OR a una porta AND a $\overline{p}$ ingressi: essa è un'uscita del circuito

Sintesi di circuiti: algoritmo duale

Il circuito precedente era un OR di p porte AND. Lo stesso problema può essere risolto dualmente mediante un AND di $\overline{p} = 2^k - p$ porte OR.

1. considero tutte le $\overline{p}$ righe della tabella di verità con uscita 0
2. per ciascuna di esse prendo una porta OR a k ingressi che restituisca 0; allo scopo applico una porta NOT su ogni ingresso che vale 1
3. invio le uscite delle $\overline{p}$ porte OR a una porta AND a $\overline{p}$ ingressi: essa è un'uscita del circuito
4. ripeto i passi 1,2,3 per ognuna delle m uscite.

Dualità delle porte AND e OR

Le due versioni appena viste sono declinazioni di un unico algoritmo, in cui abbiamo scambiato AND con OR e 0 con 1.

Si noti: una porta AND equivale logicamente a una porta OR in cui abbiamo scambiato VERO con FALSO: E = A AND B coincide logicamente con (NOT E) = (NOT A) OR (NOT B).

Generalizziamo: dato un circuito combinatorio, ottengo il circuito duale scambiando $0 \leftrightarrow 1$ per ogni segnale e AND $\leftrightarrow$ OR per ogni porta.

N.B.!!: l'algoritmo duale NON realizza il circuito duale! Realizza invece un nuovo circuito.

Es.: si verifichi questa nota su un circuito a piacere.

Esercizio

Realizzare la seguente tabella di verità

A	B	C	E
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

mediante circuiti che adoperino le due versioni dell'algoritmo visto. Realizzare poi i due circuiti duali.

Algebra booleana

La descrizione simbolica delle porte logiche (NOT, AND, OR, . . .) può essere estesa a ogni circuito combinatorio che adopera queste porte. Si ottengono così delle espressioni algebriche che descrivono reti di porte logiche.

Queste espressioni godono di proprietà algebriche che ci permetteranno, tra l'altro, di

- verificare l'equivalenza tra due circuiti
- scegliere le porte da adoperare, e dunque
- scegliere il circuito equivalente secondo un dato criterio: minimizzare il numero di porte, minimizzare il costo delle porte, minimizzare il tipo di porte da adoperare, eccetera.

Algebra booleana

George Boole (1815-1864)
Claude Shannon (1916-2001).

L'algebra booleana formalizza asserzioni di tipo linguistico come quelle viste:

- piove e fa freddo
- piove o fa freddo
- non piove
- se piove allora non splende il sole e non nevica.

La valutazione della verità di un'espressione viene ridotta a un calcolo.

I valori adoperati nel calcolo sono solo due: vero (1) e falso (0).

Valori e operatori booleani

Significato e comportamento identico a quello delle rispettive porte logiche.

Valori

TRUE	1	t
FALSE	0	f

Operatori

NOT	$\overline{}$	$\neg$	negazione	(unario)
AND	$\cdot$	$\wedge$	prodotto logico	(binario)
OR	$+$	$\vee$	somma logica	(binario)

Notazioni diverse a seconda dell'ambito: linguaggio, logica, circuiti.

Espressioni booleane e circuiti

Un'espressione booleana descrive un circuito:

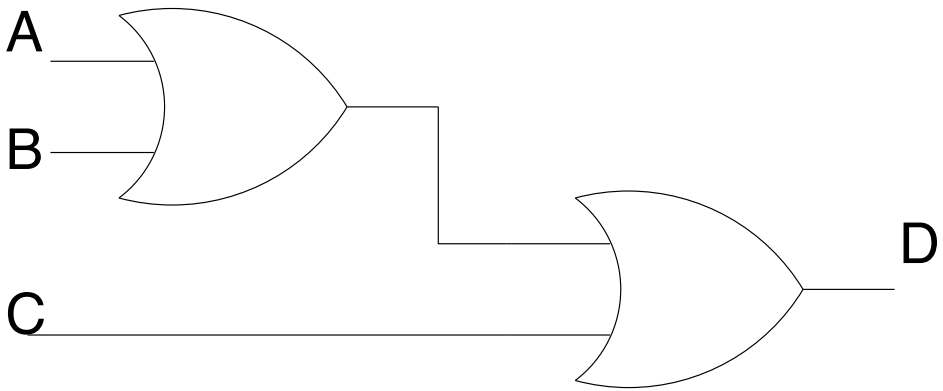
input	$\rightarrow$	argomenti (variabili)
porta logica	$\rightarrow$	operatore booleano
circuito combinatorio	$\rightarrow$	espressione booleana
output	$\rightarrow$	valore dell'espressione
segnali intermedi	$\rightarrow$	sotto-espressioni

Espressioni booleane e circuiti

Un'espressione booleana descrive un circuito:

input	$\rightarrow$	argomenti (variabili)
porta logica	$\rightarrow$	operatore booleano
circuito combinatorio	$\rightarrow$	espressione booleana
output	$\rightarrow$	valore dell'espressione
segnali intermedi	$\rightarrow$	sotto-espressioni

Es.:



$$D = (A + B) + C$$

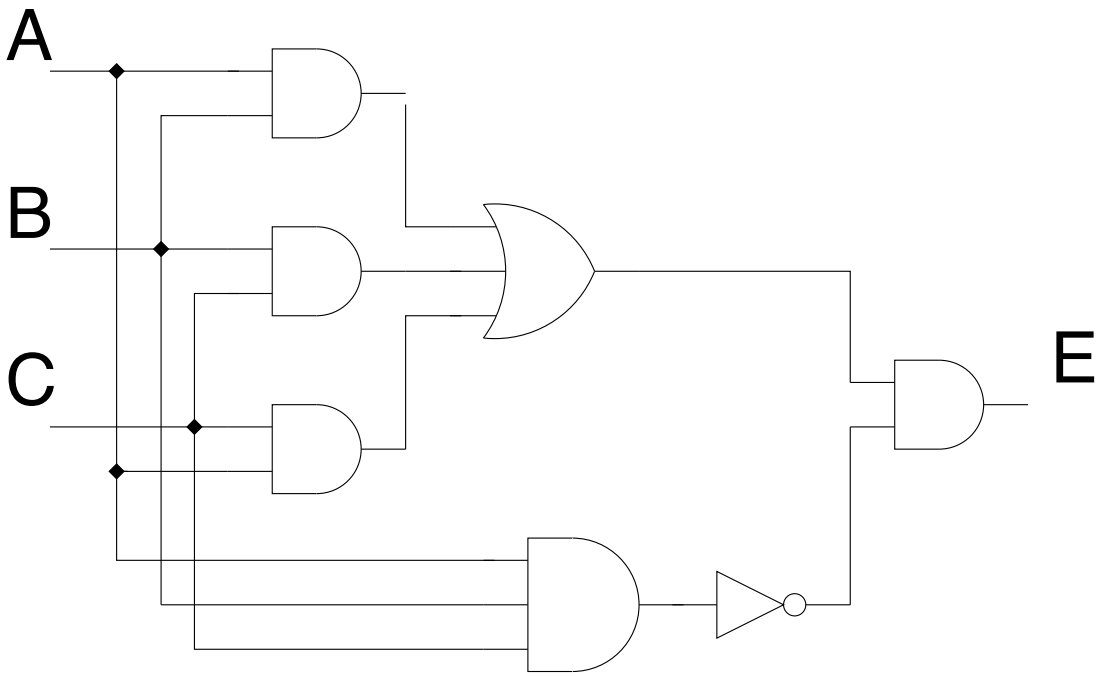
Notazione booleana

L'algebra booleana adopera le seguenti notazioni, alcune simili all'algebra numerica:

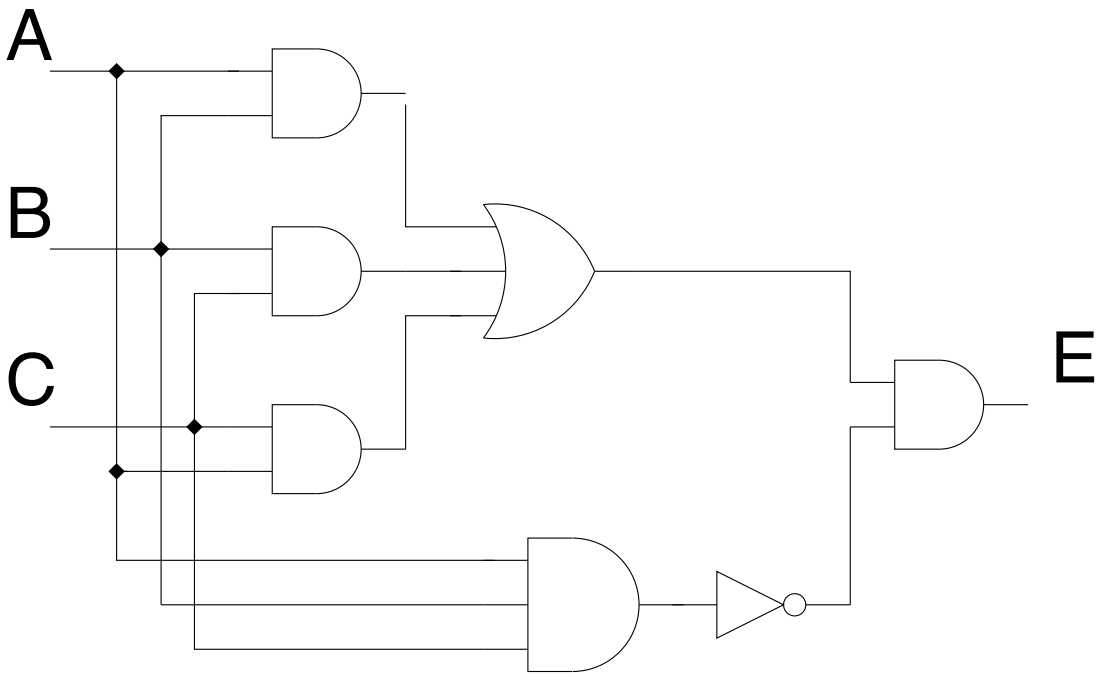
- il segno di prodotto può essere omesso: $AB = A \cdot B$
- il prodotto ha precedenza sulla somma: $A + BC = A + (B \cdot C)$
- raramente la negazione è rappresentata con $'$: $(A + B)' = \overline{A + B}$

Verificare che AND e OR sono operatori associativi:
(AB)C = ABC = A(BC)
(A + B) + C = A + B + C = A + (B + C).

Notazione booleana: esempio pratico



Notazione booleana: esempio pratico



$$E = ((A \cdot B) + (B \cdot C) + (A \cdot C)) \cdot (\overline{A \cdot B \cdot C})$$

cioè

$$E = (AB + BC + AC)\,\overline{ABC}.$$

Attenzione: $\overline{ABC} \neq \overline{A}\,\overline{B}\,\overline{C}$!!!

Proprietà dell'algebra booleana

Elemento nullo	$0 + A = A$
Identità	$1 + A = 1$
Idempotenza	$A + A = A$
Inverso	$A + \overline{A} = 1$
Commutatività	$A + B = B + A$
Associatività	$(A + B) + C = A + (B + C)$
Distributività	$A(B + C) = AB + AC$
Assorbimento	$A + (A \cdot B) = A$
De Morgan	$\overline{A + B} = \overline{A} \cdot \overline{B}$
Negazione	$\overline{\overline{A}} = A$

Dualità e complementarietà

La dualità dei circuiti combinatori si riflette in una proprietà di dualità dell'algebra booleana.

Data un'espressione booleana E, la sua negata $\overline{E}$ si ottiene negando le variabili, scambiando le operazioni $+$ e $\cdot$ e scambiando le costanti 0 e 1.

Attenzione: l'espressione $\widetilde{E}$ ottenuta scambiando tra loro solo le operazioni $+$ e $\cdot$ e le costanti 0 e 1 complementa (non dualizza!) il risultato $\overline{E}$.

- $E = A + \overline{A} \cdot (B + 0)$
- $\overline{E} = \overline{A} \cdot (A + \overline{B} \cdot 1)$
- $\widetilde{E} = A \cdot (\overline{A} + B \cdot 1) \neq E \neq \overline{E}$.

Proprietà duali

Identità	$1 \cdot A = A$
Elemento nullo	$0 \cdot A = 0$
Idempotenza	$A \cdot A = A$
Inverso	$A \cdot \overline{A} = 0$
Commutatività	$A \cdot B = B \cdot A$
Associatività	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
Distributività	$A + BC = (A + B)(A + C)$
Assorbimento	$A \cdot (A + B) = A$
De Morgan	$\overline{A \cdot B} = \overline{A} + \overline{B}$
Negazione	$\overline{\overline{A}} = A$

Considerazioni sulle proprietà

- Identità, commutatività, associatività, distributività (somma sul prodotto), elemento nullo, sono valide anche per l'algebra numerica Le altre no.
- Tutte le equivalenze dell'algebra degli interi restano valide, tenuto conto che la negazione non ha nessuna attinenza con le operazioni di opposto o inverso.
- Le proprietà viste sono dimostrabili mediante tabelle di verità, mediante argomenti di logica. Alcune conseguono da altre proprietà.

Es.: dimostrare le proprietà di assorbimento.

Semplificazione di espressioni

- Identità ed elemento nullo permettono di eliminare le costanti 0 e 1 dalle espressioni.
- Commutatività, associatività, inverso, idempotenza e assorbimento permettono di riordinare variabili ed eliminare doppioni.
- Le regole di De Morgan e permettono di separare negazioni di variabili.
- La proprietà distributiva permette di riscrivere un'espressione come somma di prodotti (forma normale). Controparte algebrica: polinomio.

Es.: si verifichi che i circuiti combinatori realizzati dall'algoritmo di sintesi (prima versione) sono descritti da forme normali.

Esempi di semplificazione

Attraverso le regole indicate si possono semplificare le seguenti espressioni.

- Identità: $A \cdot (1 + B) + 1 \cdot A + B$

Esempi di semplificazione

Attraverso le regole indicate si possono semplificare le seguenti espressioni.

- Identità: $A \cdot (1 + B) + 1 \cdot A + B$
- De Morgan: $\overline{A \cdot B + \overline{C}}$, $\overline{A \cdot B \cdot C}$, $\overline{A + B + C}$.

Esempi di semplificazione

Attraverso le regole indicate si possono semplificare le seguenti espressioni.

- Identità: $A \cdot (1 + B) + 1 \cdot A + B$
- De Morgan: $\overline{A \cdot B + \overline{C}}$, $\overline{A \cdot B \cdot C}$, $\overline{A + B + C}$.
- Distributiva: $((A + B) \cdot C + A) \cdot \overline{B}$

Esempi di semplificazione

Attraverso le regole indicate si possono semplificare le seguenti espressioni.

- Identità: $A \cdot (1 + B) + 1 \cdot A + B$
- De Morgan: $\overline{A \cdot B + \overline{C}}$, $\overline{A \cdot B \cdot C}$, $\overline{A + B + C}$.
- Distributiva: $((A + B) \cdot C + A) \cdot \overline{B}$
- Più regole: $\overline{A \cdot \overline{B \cdot C}} \cdot \overline{A \cdot B \cdot C}$

Esempi di semplificazione

Attraverso le regole indicate si possono semplificare le seguenti espressioni.

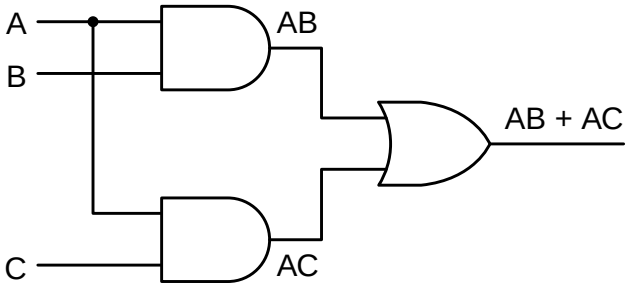
- Identità: $A \cdot (1 + B) + 1 \cdot A + B$
- De Morgan: $\overline{A \cdot B + \overline{C}}$, $\overline{A \cdot B \cdot C}$, $\overline{A + B + C}$.
- Distributiva: $((A + B) \cdot C + A) \cdot \overline{B}$
- Più regole: $\overline{A \cdot \overline{B \cdot C}} \cdot \overline{A \cdot B \cdot C}$
- Più regole: $(A + B) \cdot \overline{(A \cdot \overline{B} \cdot \overline{0})} \cdot \overline{(A \cdot B \cdot C)}$

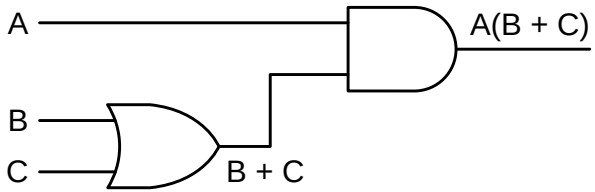
N.B.: semplificare un'espressione non ha sempre come controparte la realizzazione un circuito equivalente che adopera un numero minore di porte.

Espressioni $\subset$ Tabelle verità

Una tabella di verità si ottiene da più espressioni, e quindi è realizzata da più circuiti.

Es.: $AB + AC = A(B + C)$





A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

(a)

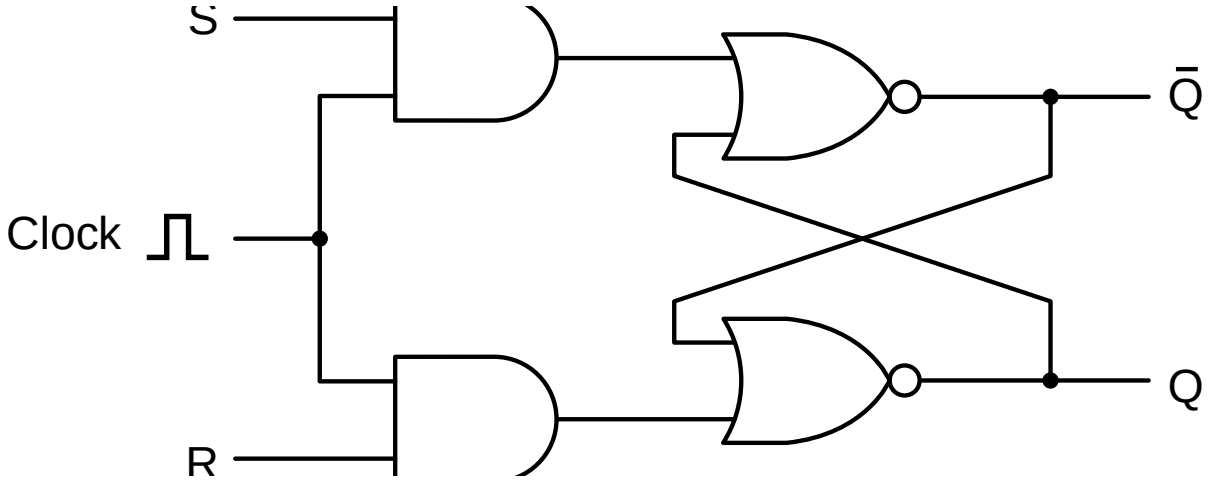
A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

(b)

Circuiti $\subset$ Espressioni $\subset$ Tabelle verità

Alcuni circuiti non possono essere descritti con un'espressione booleana:

- circuiti in cui un'uscita comanda più ingressi
- circuiti con retroazione



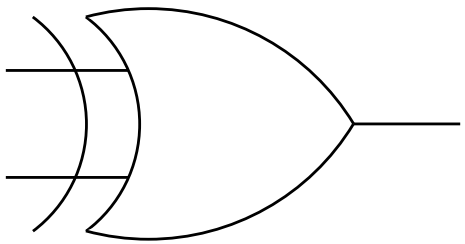
C'è maggiore libertà nel creare collegamenti tra porte logiche che nello scrivere espressioni.

Porte OR esclusivo

Essendo adoperate, per comodità si definiscono anche le porte OR esclusivo: XOR e NXOR.

Sono porte che escludono la contemporanea verità di tutti gli ingressi.

A	B	A XOR B	A NXOR B
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1



Espressioni di altre porte logiche

Porte logiche diverse da NOT, AND, OR sono subito descritte adoperando espressioni booleane:

$$A \text{ NAND } B = \overline{A \cdot B} = \overline{AB}$$
$$A \text{ NOR } B = \overline{A + B}$$
$$A \text{ XOR } B = A\overline{B} + \overline{A}B = (A + B) \cdot (\overline{A} + \overline{B})$$
$$A \text{ NXOR } B = \overline{(A + B) \cdot (\overline{A} + \overline{B})} = AB + \overline{A}\,\overline{B}$$

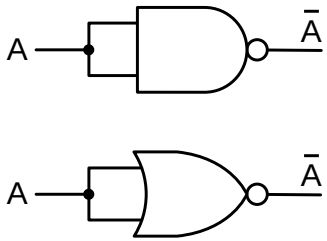
Anche le porte NAND e NOR accettano p ingressi:

$$\text{NAND}(A_1, \ldots, A_p) = \overline{A_1 \cdots A_p}$$
$$\text{NOR}(A_1, \ldots, A_p) = \overline{A_1 + \cdots + A_p}.$$

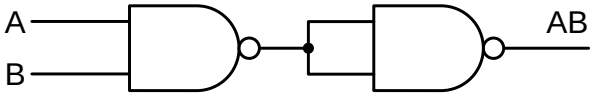
Informatici: LogiSIM definisce una porta XOR con p ingressi.

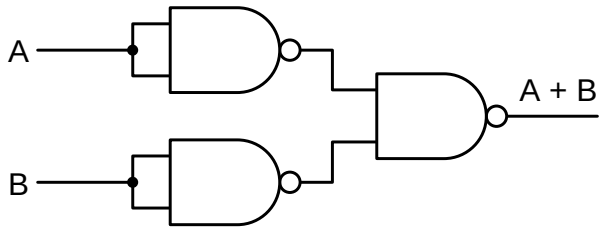
NOT, AND, OR da NAND o da NOR

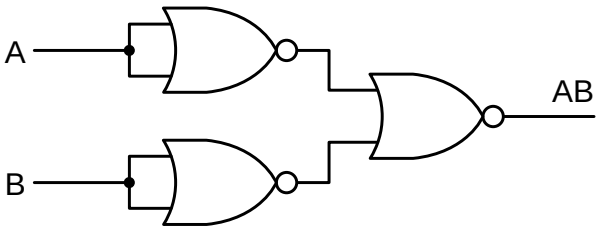
Dalle equivalenze $\overline{A} = \overline{AA} = \overline{A + A}$ discende la possibilità di ottenere le porte NOT, AND, OR adoperando solo porte NAND o porte NOR:

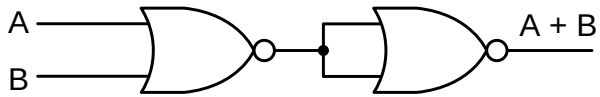


(a)









Completezza di NAND e di NOR

In generale, non è vero che NOT, AND, OR sono insostituibili. Un'espressione in forma normale può essere sempre realizzata con porte NAND:

$$A_1 \cdot A_2 \cdot A_3 + B_1 \cdot B_2 + C_1 \cdot C_2 \cdot C_3 \cdot C_4 =$$

$$\overline{\overline{A_1 \cdot A_2 \cdot A_3 + B_1 \cdot B_2 + C_1 \cdot C_2 \cdot C_3 \cdot C_4}} =$$

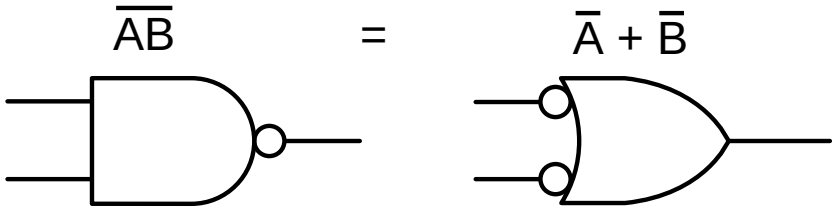
$$\overline{\overline{A_1 \cdot A_2 \cdot A_3} \cdot \overline{B_1 \cdot B_2} \cdot \overline{C_1 \cdot C_2 \cdot C_3 \cdot C_4}}$$

Nei circuiti integrati vengono adoperate quasi sempre NOT, NAND o NOR: contenendo meno transistor, sono più economiche da realizzare.

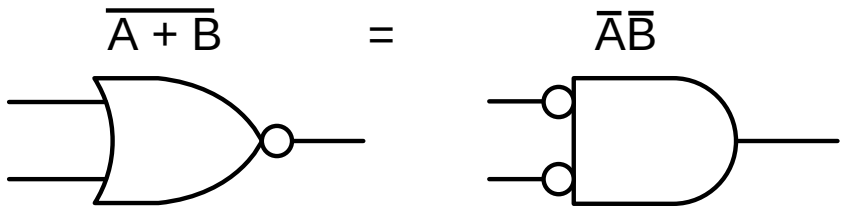
Esercizio: da una tabella di verità, come si realizza un circuito adoperando solo porte NOR?

Utilizzo della “bolla” di negazione

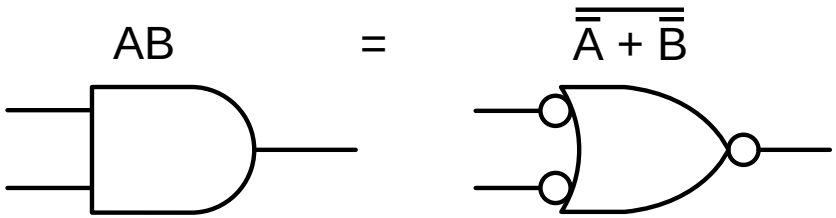
Il simbolo grafico della “bolla” di negazione può apparire anche all’ingresso delle porte logiche



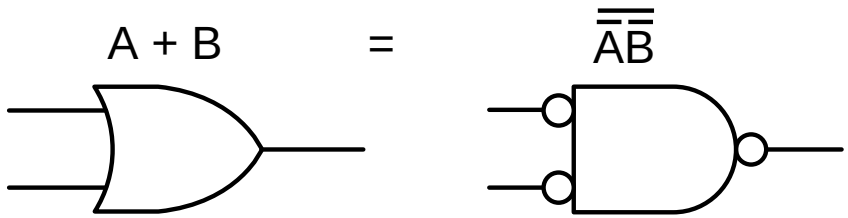
(a)



(b)



(c)



(d)

Esercizi

Semplificare le espressioni

- $A + \overline{A}B$
- $\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + AB\overline{C}$
- $\overline{A} + \overline{\overline{B} + \overline{C}} + ABC$
- $\overline{ABC} + A\overline{BC} + \overline{AB}C + AB\overline{C}$

Algebricamente e (solo per gli INF) attraverso le mappe di Karnaugh.

Mappe di Karnaugh (solo Informatici)

L'algoritmo di sintesi in generale realizza circuiti con più porte logiche del necessario.

Le mappe di Karnaugh sintetizzano circuiti a minimo numero di porte logiche. Si basano sulla regola seguente: una porta OR che riceve segnali da porte AND con un ingresso comune, negato e non, può essere rimosso e sostituito da un'unica porta AND senza quell'ingresso.

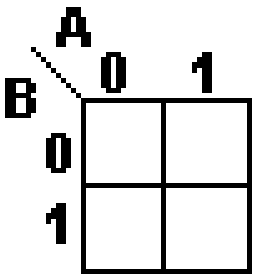
(A AND B AND C) OR (A AND (NOT B) AND C) = A AND C .

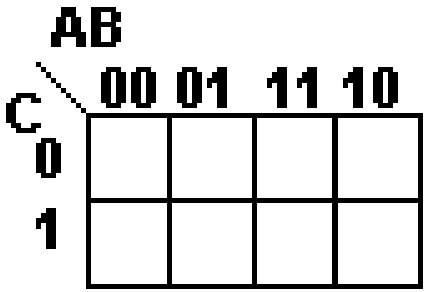
La regola può essere estesa a più ingressi:

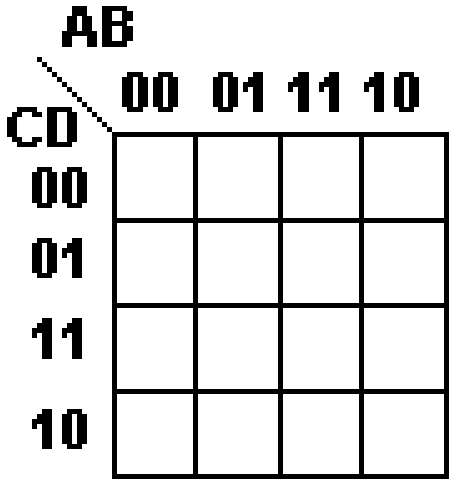
(A AND B AND C) OR (A AND (NOT B) AND C) OR ((NOT A) AND B AND C) OR ((NOT A) AND (NOT B) AND C) = (A AND C) OR ((NOT A) AND C) = C.

Matrici di tabelle di verità

Riscriviamo le tabelle di verità in forma di matrici:





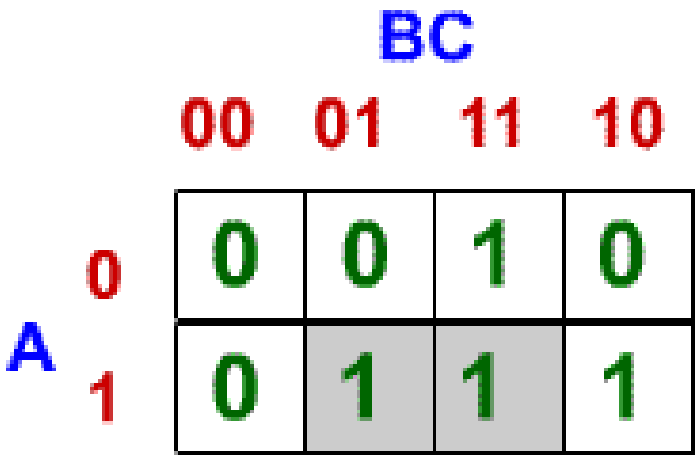


Notare l'ordine di scrittura delle combinazioni: 00, 01, 11, 10. Cioè: righe e colonne adiacenti differiscono ciascuna per un valore di segnale.
In tal modo le mappe visualizzano ogni eventuale adiacenza (X AND Y) OR (X AND (NOT Y)).

Primo esempio

Adoperiamo la notazione compatta:
A AND $B = AB$, A OR $B = A + B$, NOT $A = \overline{A}$.

Es.: realizzare $E = A\overline{B}C + ABC + \overline{A}BC + AB\overline{C}$



$$A\overline{B}C + ABC = AC$$

Raggruppando anche le adiacenze su A e su C:

$E = AC + BC + AB$.

Raggruppamenti di valori adiacenti

In generale bisogna raggruppare le adiacenze di uscite uguali a 1 che appaiono in un rettangolo (anche quadrato) di 2, 4, 8, . . . , (2^n) elementi.

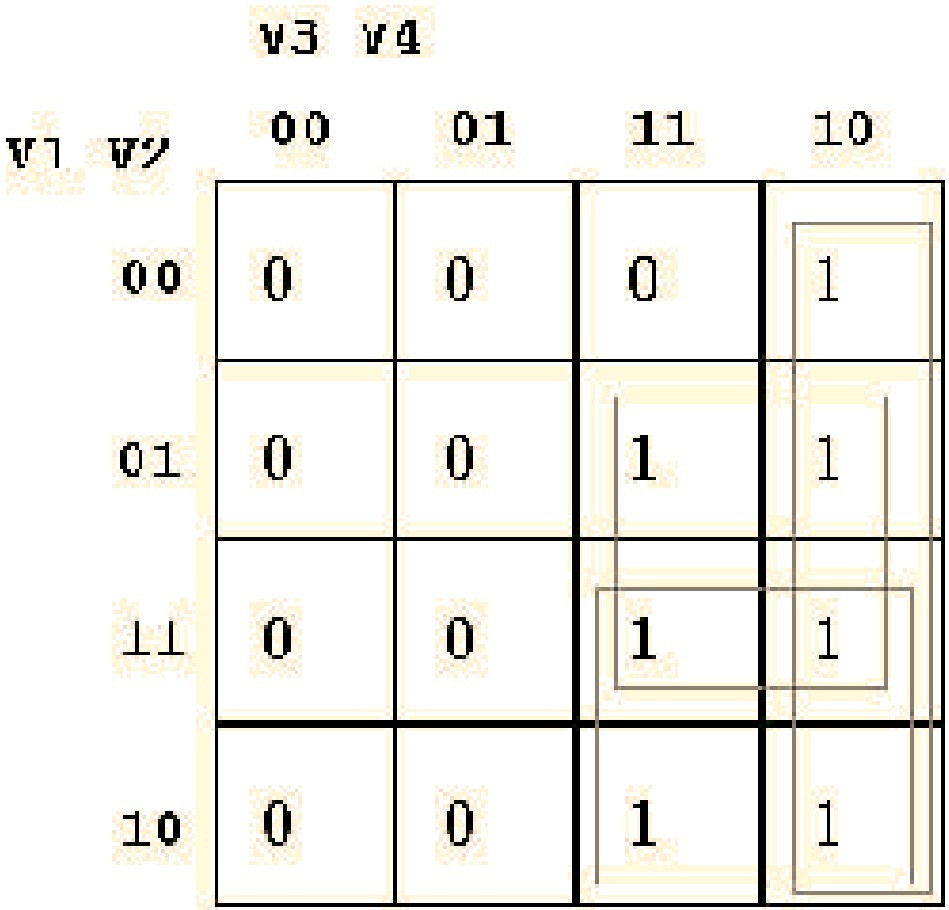
In tal modo si esaurisce ogni rettangolo, parallelepipedo, iper-rettangolo, le cui variabili assumono sempre il valore 1.

Ciò corrisponde sempre a eliminare le variabili che compaiono asserite e negate nel rettangolo, ovvero nel corrispondente OR di AND.

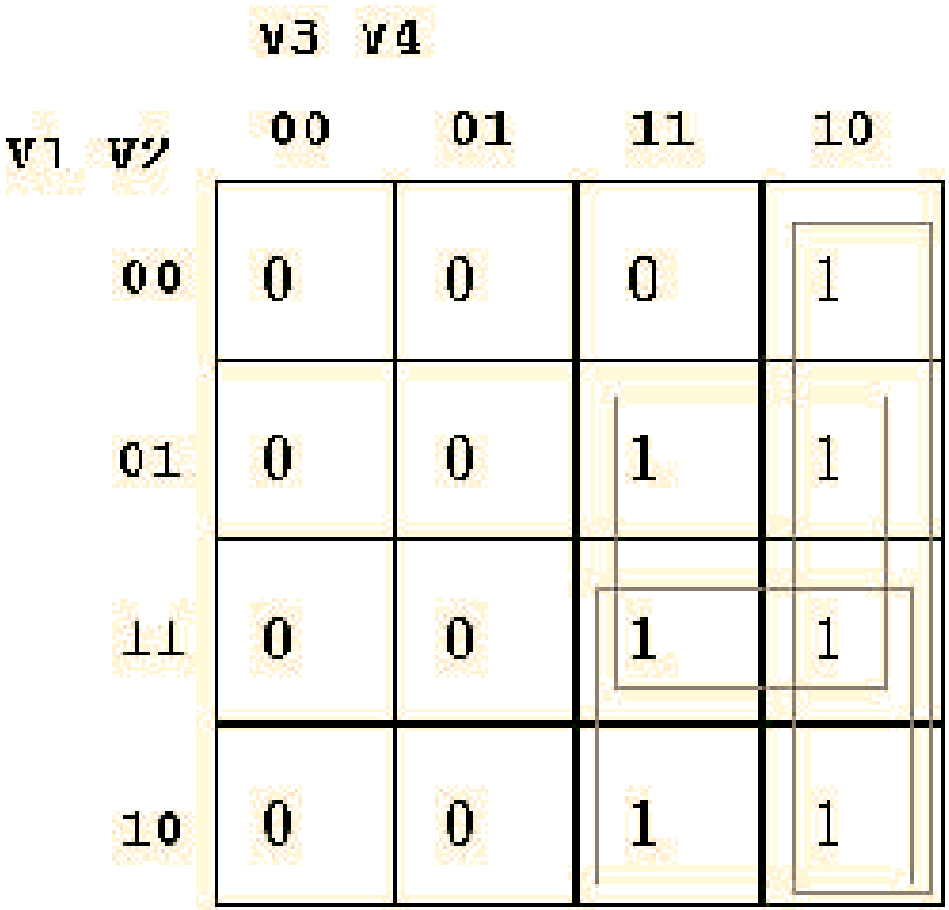
Es. (rettangolo 2×2 di 1 in una mappa 4×4):

$$ABCD + \overline{A}BCD + ABC\overline{D} + \overline{A}BC\overline{D} = BCD + BC\overline{D} = BC.$$

Mappa 4 $\times$ 4 con rettangoli di 4 “uni”



Mappa 4 $\times$ 4 con rettangoli di 4 “uni”



$$E = V_3\overline{V_4} + V_2V_3 + V_1V_3$$

Adiacenze sui lati opposti

Per come sono definite le matrici, anche i lati opposti sono adiacenti.

WX \ YZ	00	01	11	10
00	1	0	0	1
01	1	1	0	0
11	1	0	0	0
10	1	0	0	1

Adiacenze sui lati opposti

Per come sono definite le matrici, anche i lati opposti sono adiacenti.

WX \ YZ	00	01	11	10
00	1	0	0	1
01	1	1	0	0
11	1	0	0	0
10	1	0	0	1

$$E = \overline{Y}\,\overline{Z} + \overline{X}\,\overline{Z} + \overline{W}X\overline{Y}$$

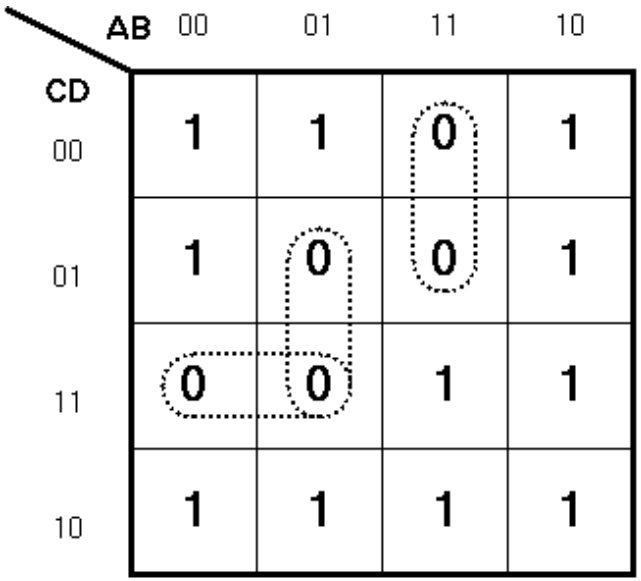
Karnaugh: applicazione duale

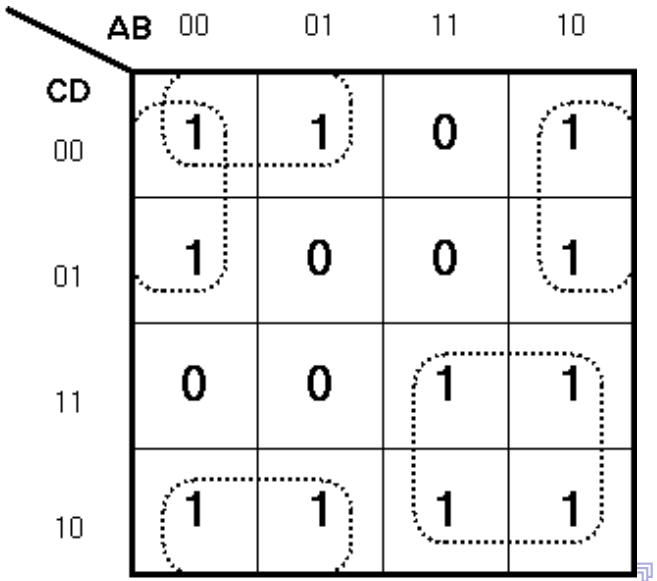
AB + A$\overline{B}$ = A. Dualizzando: ($\overline{A}$+$\overline{B}$)($\overline{A}$+B) = $\overline{A}$.

Applicazione standard: $E = AC + \overline{B}\,\overline{C} + \overline{A}\,\overline{D}$.

Analogamente all'algoritmo duale di sintesi, raggruppiamo gli zeri in un AND di OR di variabili in cui neghiamo gli ingressi uguali a uno:

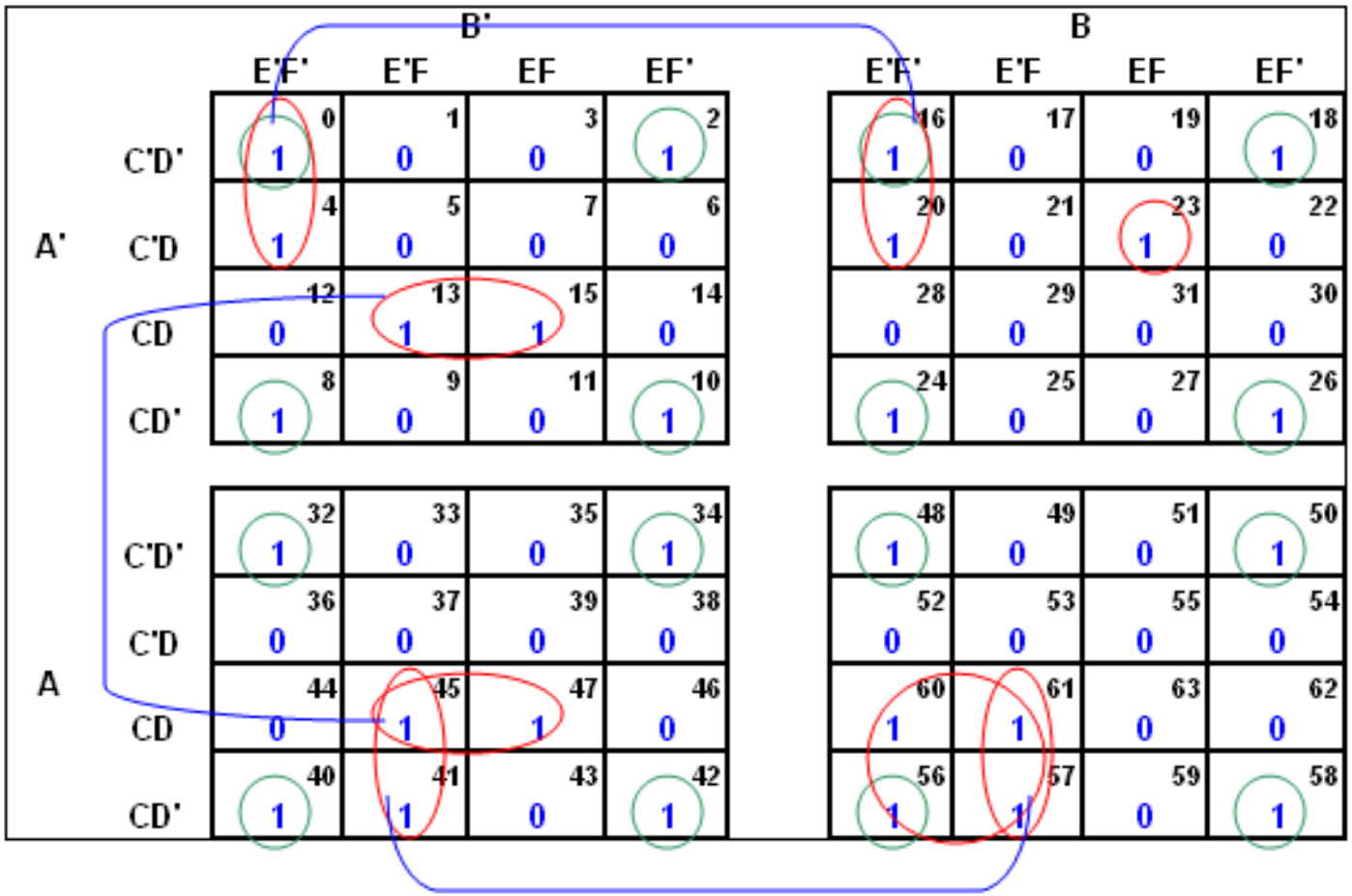
$$E = (\overline{A} + \overline{B} + C)(A + \overline{B} + \overline{D})(A + \overline{C} + \overline{D})$$





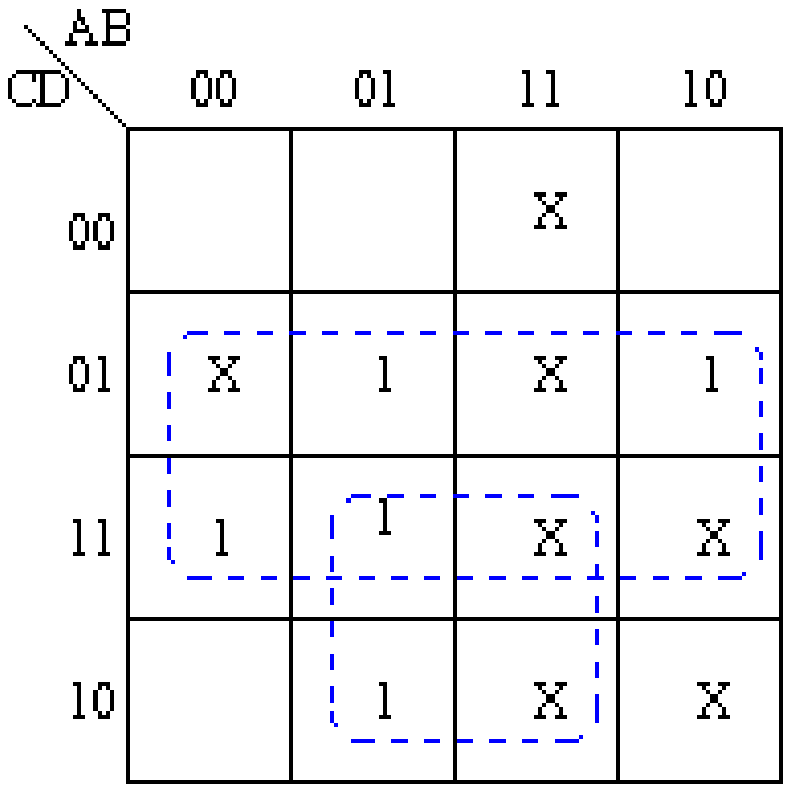
Mappe di Karnaugh con 5 o 6 variabili

Si rendono i parallelepipedi disegnando 2 o 4 mappe di Karnaugh a 4 variabili: caselle corrispondenti in mappe adiacenti sono considerate adiacenti.



Presenza di uscite indeterminate

Le mappe di Karnaugh evidenziano immediatamente la scelta da fare sui valori indeterminati nell'uscita al fine di minimizzare il numero di porte nel circuito.



$$E = BC + D.$$