24 Marzo 2023

ARCHITETTURE DI CALCOLO LEZIONE 6

Reti combinatorie

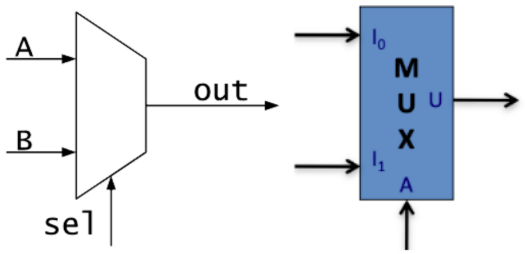
Esistono infiniti circuiti logici in grado d'implementare una tavola di verità, per cui bisogna scegliere il più efficiente. I due parametri di valutazione da considerare sono la complessità e la velocità, dipendenti dal numero delle porte e dal percorso più lungo dell'input per raggiungere l'output.

Espressioni canoniche

- La I forma canonica (Somma di prodotto SP): ogni funzione di n variabili binarie è descritta da una somma di tanti termini prodotto quante sono le configurazioni per cui la funzione stessa vale 1. In ogni mintermine (termine prodotto) compaiono tutte le n variabili, ciascuna in forma vera se nella configurazione corrispondente vale 1, altrimenti in forma negata.
- La II forma canonica (Prodotto di somma PS): ogni funzione di n variabili binarie è descritta da un prodotto di tanti termini somma quante sono le configurazioni per cui la funzione stessa vale 0. In ogni maxtermine (termine somma) compaiono tutte le n variabili, ciascuna in forma vera se nella configurazione corrispondente vale 0, altrimenti in forma negata.

Usando PS o SP si scrive sempre un circuito corretto, in quanto i circuiti risulteranno equivalenti per numero di porte logiche e complessità ma avranno funzionamento esattamente opposto.

Multiplexer



I circuiti possono essere combinatori o sequenziali.
La tipologia più semplice di circuito combinatorio è il multiplexer, o selettore (abbreviato MUX), rappresentato dai due schemi a blocchi di fianco.
In elettronica, il termine può riferirsi a una tipologia di circuiti integrati, oppure a una particolare apparecchiatura completa.
Nel primo caso un **multiplatore** (anche chiamato **selettore**) è un dispositivo capace di selezionare un singolo segnale elettrico fra diversi segnali in ingresso in base al valore degli **ingressi di selezione**.
Esistono multiplatori sia per segnali digitali sia per segnali analogici (**amux**).
Un multiplexer permette di selezionare uno degli n ingressi che sarà poi presentato in uscita, grazie a log2^n linee di comando (ldc) (es. 4 ingressi→2 ldc, 8 ingressi→3 ldc,

7 ingressi→3 ldc). Per ottenere un numero maggiore di combinazioni, i multiplexer possono essere collegati in serie.
Facendo riferimento all'immagine in blu, se A=0→l'output = bit di I_0, se A=1 →l'output = bit di I_1.
Una volta creata la tavola di verità del multiplexer, è possibile ricavarne l'espressione canonica SP dell'uscita.
In generale, il circuito di un multiplexer prevede sempre un OR che prende in input tante AND quanti sono gli ingressi; le AND a loro volta, ricevono ciascuna un ingresso e una linea di controllo, in forma vera o negata.
Un multiplexer, quindi, può essere rappresentato da un circuito a due livelli (si escludono dal calcolo i NOT); tuttavia, in casi più complicati dove vi sono 16, 32 o più ingressi, sarà necessario ricorrere all'uso di ulteriori OR, con conseguente aumento dei livelli.
Il circuito inverso del multiplexer prende il nome di demultiplexer, di solito abbiamo uno di entrambi, lo scopo è quello di riuscire a scegliere l'output in base agli input che arrivano.

Tabella della verità

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

*Espressione **canonica** SP dell'uscita*

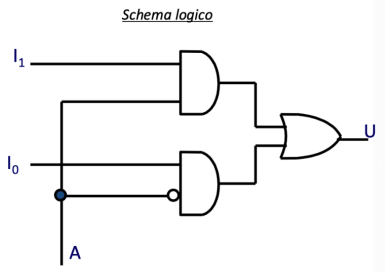
$$U = \bar{A}\bar{I}_1 I_0 + \bar{A}\, I_1 I_0 + A\, I_1 \bar{I}_0 + A\, I_1 I_0$$

$$U = \bar{A}\, \bar{I}_1\, I_0 + \bar{A}\, I_1 I_0 + A\, I_1\, \bar{I}_0 + A\, I_1 I_0 =$$

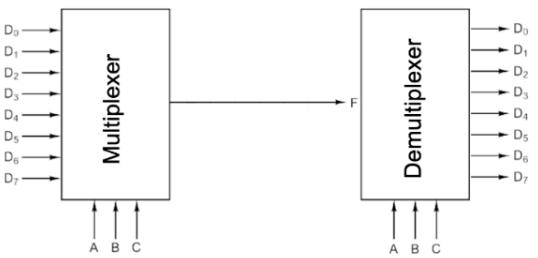
$$= \bar{A}\, I_0\, (I_1 + \bar{I}_1) + A\, I_1\, (I_0 + \bar{I}_0) =$$

$$= \bar{A}\, I_0 + A\, I_1$$

Schema logico



Demultiplexer
Il circuito inverso del multiplexer è il demultiplexer (indicato con la sigla DEMUX), il quale riceve 1 input dati (F) e lo trasmette ad una delle 2^n linee di output (D), dove n è il numero di input di controllo (in questo caso sono 3→A, B, C).
A seconda del valore che assume quest'ultimo, verrà attivata una specifica linea di output (es. A=0, B=0, C=0→si attiva D0; A=0, B=0, C=1 →si attiva D1). Il demultiplexer, in combinazione con una linea dati e un multiplexer va a costituire la struttura che si cela dietro i combinatori telefonici ed i



centralini telefonici (figura). Infatti, essendo impossibile installare [n(n-1)]/2)] cavi telefonici per le n persone da mettere in collegamento, furono ideati i centralini telefonici, prima gestiti manualmente dai centralinisti e, in seguito, automatizzati.
Il dispositivo complementare, il **deselettore** o **demultiplatore** (ingl. **demultiplexer,** abbr. **demux**), ha un solo ingresso e diverse uscite.
Un **demultiplatore** è un circuito logico la cui principale funzione è inversa a quella del **multiplatore**.
Esso è quindi una rete combinatoria con k ingressi (di selezione) e m = 2^k uscite, ciascuna delle quali è attiva soltanto in corrispondenza di uno dei 2^k valori di ingresso.
In base al valore degli ingressi di selezione, l'ingresso viene collegato a una delle uscite. Per esempio, un demultiplatore a otto uscite ha un segnale di ingresso (X), tre ingressi di selezione (S2, S1 e S0), e otto uscite (da A0 a A7). Se per esempio S2 e S0 sono a '1' e S1 è a '0', l'uscita A5 sarà uguale a X e tutte le altre uscite saranno messe a 0. Il demultiplexer ha la funzione esattamente inversa al multiplexer: il multiplexer infatti riunisce più entrate in un'unica uscita mentre il demultiplexer smista un ingresso in più uscite.

Decoder
Il decoder o decodificatore, prende due parametri n ed m, esso riceve in ingresso n bit e presenta in uscita 2^n bit, in sostanza asserisce(da valore 1) solamente le linee con valore binario di interesse, per cui es. X0=0, X1=0→si attiva Y0; X0=0, X1=1→si attiva Y1 quindi la cella con la posizione corrispondente a quel valore.
L'uscita del decodificatore quindi è un insieme di 0 tranne una posizione in cui troviamo l'1 che rappresenta l'espressione del numero binario di ingresso.
I decodificatori hanno dei bit chiamati di abilitazione settati a 0 affinché poi sia corretta l'uscita che viene valutata dal decoder stesso.
Un decoder può avere anche un ingresso di ENABLE (sigla: EN*; * indica che è un tipo di abilitazione a valore basso), che deve essere settato a 0 se si vuole che l'output sia abilitato.
I segnali di abilitazione si suddividono in due famiglie, a valore basso che quindi hanno valore 0 oppure a valore alto e quindi con valore 1.
La tabella di verità dovrà avere tante colonne a sinistra quanti sono gli ingressi più il valore del segnale di abilitazione.
Gli output saranno di numero pari alle possibili risposte che si possono avere quindi 2^n (in questo caso il prof fa l'esempio di 2 ingressi più 1 segnale di abilitazione e quindi si avranno 4 output [in quanto 2^2 poiché il segnale di abilitazione non viene contato]).
Questo è un sistema utile quando si deve identificare una sola uscita utile che quindi deve essere attivata mentre le altre devono essere spente, quindi quando avrò n linee dovrò sceglierne una sola dovrò inviare al decodificatore un codice binario che rappresenta proprio quella linea che voglio accendere, ed e qui che si torna al concetto

del logaritmo affrontato prima, cioè se io ho 4 uscite dovrò utilizzare due input che in binario rappresentano le 4 uscite.
Dopo aver scritto le espressioni canoniche SP delle uscite (che saranno tante quanti sono gli output, in questo caso 4), è possibile realizzare lo schema logico del circuito; esso avrà tante "and" quanti sono gli output, ed ognuna di esse riceverà gli input X + EN* (negati o meno).

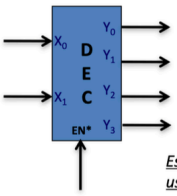
Se osserviamo la tabella di verità ,quando è che y0 vale 1 ? … solo in un caso ovvero quando i due input e anche il segnale di abilitazione varranno 0; se guardiamo la parte bassa capiamo anche che indipendentemente dal valore degli input avremo come valore di ritorno 0 quindi non verrà attivata nessuna uscita, questo perché il segnale di abilitazione vale 1 e quindi non vengono valutati gli input.
Quindi le uscite che si attiveranno nella parte alta della tabella corrispondono esattamente al valore espresso in binario dagli input (es. x1 e x0 = 1 allora il valore sarà 3 per cui si attiverà y3).
Una volta che abbiamo la tabella di verità possiamo quindi formare il circuito equivalente, prima andremo a scrivere le forme canoniche che saranno una per ogni output, concentriamoci ora su y0, esso sarà semplicemente un termine in cui tutti gli input valgono 0 ovvero (!x0) • (!x1) •(!en), y1 sarà (x0) • (!x1) •(!en), y2 (!x0) • (x1) •(!en) mente y3 sarà (x0) • (x1) •(!en); queste saranno le 4 espressioni che dobbiamo utilizzare per formare il circuito.
L'impulso di abilitazione può anche non essere utilizzato, ma di norma è presente perché questi circuiti sono integrati in ambiti più ampi in cui solo in determinati momenti devono essere attivati, quindi è importante avere la capacità di modulare la possibilità o meno di avere un'attivazione del circuito stesso; di solito si usa la convenzione dell'impulso di abilitazione a valore basso ovvero l'output viene generato se e solo se l'en vale 0 per cui nel caso l'en dovesse valere 1 indipendentemente dai valori di input l'output non viene generato perché il decodificatore ignora questi ultimi; ciò ci consente quindi di fare cosa ? … di modulare lo spegnimento delle uscite poiché se non ci fosse l'input di abilitazione obbligatoriamente una delle quattro uscite (in questo caso) rimarrebbe accesa.
Quando abbiamo un circuito che non produce un uscita sola ma produce più uscite, le cose non sono diverse ma dobbiamo semplicemente non una sola funzione di uscita ma n funzioni però il meccanismo è sempre lo stesso, basta che ci concentriamo sulle singole colonne e troviamo la funzione di uscita per quella colonna.
Perché queste espressioni sp sono costituite da un solo termine ciascuno ? perché non ci sono somme ? perché ciascuna delle funzioni di uscita vale uno solamente su una riga, ci vorrebbero almeno due uno per generare la somma, in questo caso stiamo solamente generando en questo asterisco non è un prodotto ma sta solo a rappresentare la variabile che si chiama en*.
Data le funzioni che abbiamo definito possiamo facilmente implementare il circuito decoder, quindi avremo: 4 uscite Y0, Y1, Y2 e Y3; i tre ingressi X0, X1 e en e ovviamente si tratta di un circuito a valore basso, quindi dovremmo avere per ogni uscita una and e per ogni ingresso tre input che possono essere diretti o negati.
Iniziamo con Y3 che è il prodotto di tutti e tre negati, quindi prendiamo i tre valori, li mandiamo negati nella and e poi il risultato sarà Y3, Y1 prende invece negati en e X1 e il diretto del X0 e li manda nella and.
Il circuito decoder è molto semplice, di fatto una volta capito il meccanismo non dobbiamo passare nemmeno per le formule per implementarlo, poiché richiede una and per ogni linea di uscita, poi prendere i valori di ingresso e andarli a mettere negati o diretti e sostanzialmente si capisce, questi not non fanno altro che rappresentare i valori negati.

Esempio: decoder (2:4)

Tabella della verità

EN*	X_1	X_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	1
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	0

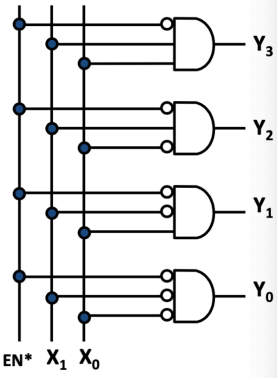
Schema a blocchi



*Espressioni **canoniche** SP delle uscite*

$Y_3 = \overline{EN*}\, X_1 X_0$
$Y_2 = \overline{EN*} X_1 \overline{X}_0$
$Y_1 = \overline{EN*}\, \overline{X}_1 X_0$
$Y_0 = \overline{EN*} \overline{X}_1 \overline{X}_0$

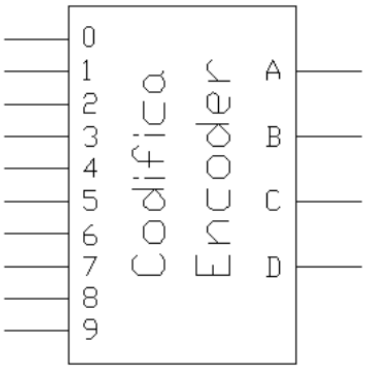
Schema logico



Encoder

Il circuito inverso del decoder è detto encoder (o codificatore), il quale permette di trasformare una posizione (ovvero l'input con valore 1) in un numero binario (es. la linea di input 5 è attiva→A=0, B=1, C=0, D=1; la linea di input 8 è attiva→A=1, B=0, C=0, D=0). Il numero di linee di output è pari a $\lceil \log_2 n \rceil$, dove n è il numero di linee di input.

Come esiste il demultiplexer che è il contrario del multiplexer, esiste l'encoder che è il contrario del decoder.
In sostanza quello che fa l'encoder è ricevere in ingresso un valore pari a 1 solo su una di queste linee, quindi riceve n-1 valori pari a 0 e solo un valore pari a 1, e tira fuori in output un numero in binario corrispondente alla posizione su cui viene messo l'1.
Se io in ingresso sulla linea 0 ho il valore 1 e su tutte le altre il valore 0, l'output corrispondente sarà il valore 0000 perché gli sto mandando il valore 1 sulla linea 0, se gli mando il valore 1 sulla seconda linea 0100, è quindi esattamente l'opposto, perché trasformo una posizione in un numero binario.
Ricordiamo che appunto il circuito decoder trasforma un numero binario in una posizione, mentre come abbiamo già ribadito prima l'encoder fa l'esatto opposto.
Qui non andiamo a generare il circuito, ma comunque è abbastanza banale da realizzare, sostanzialmente dobbiamo numerare tutti i possibili ingressi, che sono ovviamente, una matrice con tutti i valori pari a 0 tranne sulla diagonale dove ci sono i valori pari a 1; mentre il numero delle uscite sono logaritmi in base due del numero degli ingressi, approssimato.



Ingressi										Uscite			
0	1	2	3	4	5	6	7	8	9	D	C	B	A
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

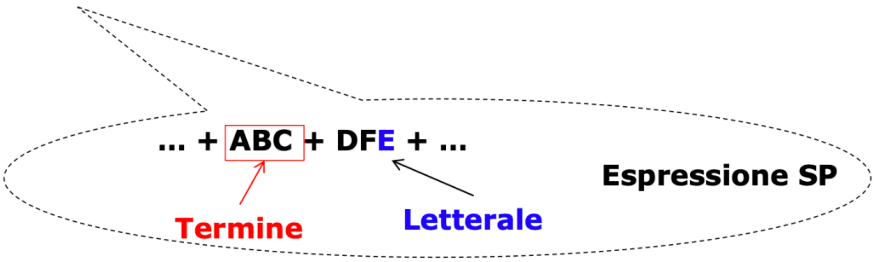
Implicante

L'implicante di una funzione booleana è il termine prodotto, che assume il valore 1 per le configurazioni della funzione in cui la funzione stessa vale 1.
Denominiamo implicato della funzione il prodotto di un certo numero di variabili che valgono 1 nelle configurazioni in cui la funzione vale 1.

Implicato

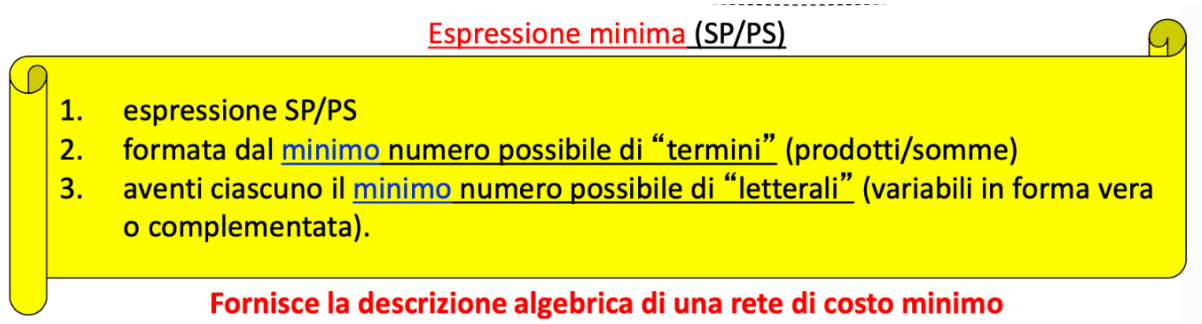
L'implicato di una funzione booleana è il termine somma di n variabili che assume il valore 0 per le configurazioni della funzione in cui essa vale 0.
Se abbiamo un espressione in prima forma canonica, ovvero in somma di prodotti, il termine non è nient'altro che uno dei termini che vengono sommati, mentre usiamo il termine letterale per identificare ciascuna delle variabili trasformate in un termine.
Quindi ad esempio nel termine DxFxE la E è un letterale, come lo è la F o la D.

Implicante primo e implicato primo
IMPLICANTE PRIMO: è un implicante da cui non possiamo togliere nessun letterale senza perdere le sue proprietà nel tempo, mentre un IMPLICATO PRIMO: è un implicato da cui non possiamo togliere nessun letterale senza che perda la sua proprietà.



Espressione minima
Un altro concetto fondamentale è l'ESPRESSIONE MINIMA, un espressione minima è un espressione irridondante (non ridondante) di implicanti o implicati primi:
- L'espressione minima di una somma SP è una somma irridondante (non possono essere eliminati termini) di implicanti primi (non possono essere eliminati letterali).
- L'espressione minima di un prodotto PS è un prodotto irridondante di implicati primi.

Quindi lo scopo è ottenere un'espressione che contenga il minor numero di termini, che a loro volta siano formati dal minor numero di variabili, senza che si alteri la funzione; questa rappresenta la descrizione algebrica di una rete di costo minimo.



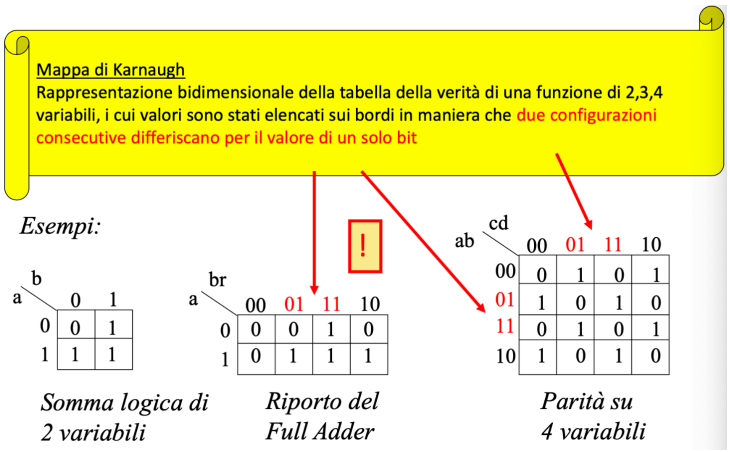
Rindondante e irridondante
IRRIDONDANTE equivale a dire non ridondante, mentre RIDONDANTE vuol dire che un espressione contiene termini di cui alcuni dei quali sono ripetitivi, quindi una espressione per essere minima vuol dire che non deve essere ridondante poiché conterrebbe alcuni termini in più che non danno valore aggiunto.
Questa nomenclatura è fondamentale ma lo è ancor di più la realizzazione di questi concetti, poiché si traducono nella realizzazione del circuito logico più sintetico, economico, veloce, efficace ed efficiente.

IRRIDONDANTE = contenuto minimo di termini nell'espressione.
PRIMO = contenuto minimo di letterali nei termini.

In questa espressione ad esempio ci sono dei termini ridondanti, lo si capisce perché quando noi li eliminiamo ci viene fuori una espressione equivalente ma irridodante.
Quindi quando parliamo di espressioni irridondanti parliamo di espressioni da cui non possiamo eliminare termini; per primo intendiamo anche che i letterali all'interno dei termini non si ripetano e che siano quindi inutili.
Se ad esempio un letterale in un termine vale sempre uno questo termine non può essere un implicante primo poiché quel letterale può essere eliminato senza che ci sia una variazione effettiva del significato.
Questo processo viene effettuato sia sulle espressioni in prima che in seconda forma canonica.
Volendo riassumere noi vogliamo un'espressione di qualunque natura sia (di prima o seconda forma canonica) che abbia al suo interno il minor numero di termini e vogliamo che questi termini siano costituiti dal minor numero di variabili.
Nel momento in cui siamo riusciti ad ottenere un espressione di questo tipo abbiamo ottenuto finalmente l'espressione minima.
L'espressione minima è quindi un espressione irridodante di implicanti primi nel caso delle somme di prodotto oppure di implicati primi nel caso di prodotti di somme.

Mappe di Karnaugh

Le mappe di Karnaugh sono rappresentazioni alternative delle tabelle di verità.
Sono uno strumento molto utile per risolvere il problema della sintesi dei circuiti, sono sostanzialmente rappresentazioni alternative delle tabelle di verità, che normalmente hanno a sinistra le variabili di input con le loro casistiche e a destra quelle di output e ciò può essere trasformato facilmente in una mappa bidimensionale di funzione che accetta fino a 6 variabili.
Le mappe di karnaugh si possono utilizzare facilmente quando il numero delle variabili è 2 oppure 4 ma in realtà si possono utilizzare ed estendere con un numero di variabili di 5 e di 6, oltre questo limite questo metodo grafico non si può più usare, in quanto diverrebbe tridimensionale e non si riuscirebbe più a gestire, anche se in realtà raramente capita, ma se dovesse succedere si ricorre ad altre metodologie.
I valori sono elencati sui bordi in maniera che due configurazioni consecutive differiscano per il valore di un solo bit (distanza di Hamming=1).
È per rispettare quest'ultimo principio che il binario "01" è seguito da "11" e non "10", come avviene quando si rispetta l'ordine crescente. Infatti, "01" e "11" differiscono di un solo bit (il primo) mentre "01" e "10" differiscono di due bit.
Inoltre, a differenza delle tavole di verità, all'output non viene assegnato alcun nome.
La prima mappa rappresenta la somma logica di 2 variabili (a, b) e funziona come una matrice: la prima casella in alto a sinistra è il risultato della somma tra la variabile a sinistra (0) e la variabile in alto (0).
Il secondo esempio è una mappa di Karnaugh a tre variabili (a, b, r), quindi rettangolare, che raffigura un circuito "Full Adder" (riceve in ingresso due bit e il riporto precedente e calcola in output la somma e il riporto successivo); in questo caso l'output è solo il riporto successivo mentre per la somma servirà un'altra mappa.
Di seguito è riportata la tabella di verità del circuito Full Adder dalla quale è più semplice ricavare la rispettiva mappa (guardare "Riporto del Full Adder" nell'immagine precedente).



Ad esempio, la prima mappa rappresentata è la somma logica di 2 variabili (a, b) e funziona come una matrice: la prima casella in alto a sinistra è il risultato della somma tra la variabile a sinistra (0) e la variabile in alto (0).
Il secondo esempio è una mappa di Karnaugh a tre variabili (a, b, r), quando abbiamo 3 variabili la mappa risultante é di forma rettangolare, in questo caso (che sta mostrando) il circuito è chiamato FULL ADDER ovvero un circuito che fa la somma di due bit di ingresso ed il riporto precedente, portando in output la somma ed il riporto successivo.
Un full adder, ovvero un addizionatore completo è un circuito che somma due variabili, quindi per effettuare la somma cosa riceve ? I due input da sommare e il riporto della somma precedente, per poi mandare in output la somma dei tre (i due input e il riporto) ed il riporto successivo, ovvero il riporto della somma effettuata che poi slitterà alla somma successiva.
Questa è la tabella di verità del circuito di somma con il riporto, e vediamo se ci ritroviamo, questa tabella che ci dice ? Che se A è pari a 0 e B è pari a 0 ed R è pari a 0 vale tutto 0, ovvero sia la somma che il riporto successivo è pari a 0; se invece A e B sono pari a 0 mentre il riporto è pari a 1, avremo come risultato la somma pari a 1 e il riporto successivo pari a 0; se A è pari 0 mentre sia B che R sono pari a 1, avremo somma pari a 0 e il riporto successivo pari a 1; se invece tutti e tre i valori, A B ed R sono pari a 1 allora la sia la somma che il riporto saranno pari a 1.
IN QUESTO CASO NEL FULL ADDER COME RISULTATO VIENE CALCOLATO IL RIPORTO NELLA CASISTICA IN ESAME !!!
SI PUÒ OVVIAMENTE FORMARE ANCHE UN FULL ADDER IN CUI VIENE RIPORTATA LA SOMMA E NON IL RIPORTO.
C'è una particolarità molto importante, visualizzabile in quanto il full adder genera come risultato due output, motivo per il quale verranno formate due mappe o tabelle separate, questo poiché si necessita di una mappa di Karnaugh separata per ogni output.
Scriviamo la tabella di verità in modo tale da capire come possiamo scrivere la mappa di karnaugh, elenchiamo quindi le combinazioni, sono 3 variabili per cui le possibili combinazioni sono 8, esse vengono riportate in ordine crescente (000, 001, 010…), l'ordine con cui scriviamo i numeri nella tabella di verità di norma è proprio quello dell'ordine crescente infatti qui non vale il discorso della minima distanza di hamming (infatti non sono a distanza 1, basta vedere il secondo e il terzo caso che si trovano a distanza 2).
Troviamo adesso somma e riporto.
Se si genera un riporto è chiaramente la cifra più significativa che va nel riporto, ma questo è un discorso più ampio in quanto non valido solo con i binari ma anche con i decimali ad esempio (es. 5+5 ci da 0 con riporto 1 quindi 10).

Ora da questa tabella di verità noi possiamo generare due mappe di karnaugh. Scriviamo l'espressione in prima forma canonica della nostra tabella, ovvero la somma di prodotti, quindi troviamo S ed r' (prendiamo le righe in cui c'è uno come risultante e prendiamo i valori interni che determinano il risultato [i valori in cui c'è 1 li prendiamo diretti mentre quelle in cui c'è 0 li prendiamo negati e se sono sulla stessa riga li moltiplichiamo tra loro]).

a	b	r	s	r'
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

Quindi questa è la tabella di verità, da questa tabella di verità noi possiamo costruire quante mappe di Karnaugh?
2 perché si genere una mappa di Karnaugh per ogni output e quindi l'esempio che vedete li chiamato riporto del full adder non è altro che la mappa di Karnaugh che viene cosrtuita da questa tabella ma considerando solamente la r', e come si genera?
Cominciamo con generare le etichette delle righe e delle colonne, banalmente sulle righe mettiamo la prima variabile e la etichettiamo 0 oppure 1 sulla colonna andiamo a mettere le altre sue variabili considerando le 4 combinazioni di valori possibili che sono 00- 01- 11 -10 e vi ripeto, perché da 01 si passa ad 11 e non a 10 perché la mappa di Karnaugh funziona solo se ogni cella cambia etichetta dalla successiva al massimo per un bit, poi una volta che abbiamo costruito questo dobbiamo semplicemente andare a copiare li dentro i valori di r' in

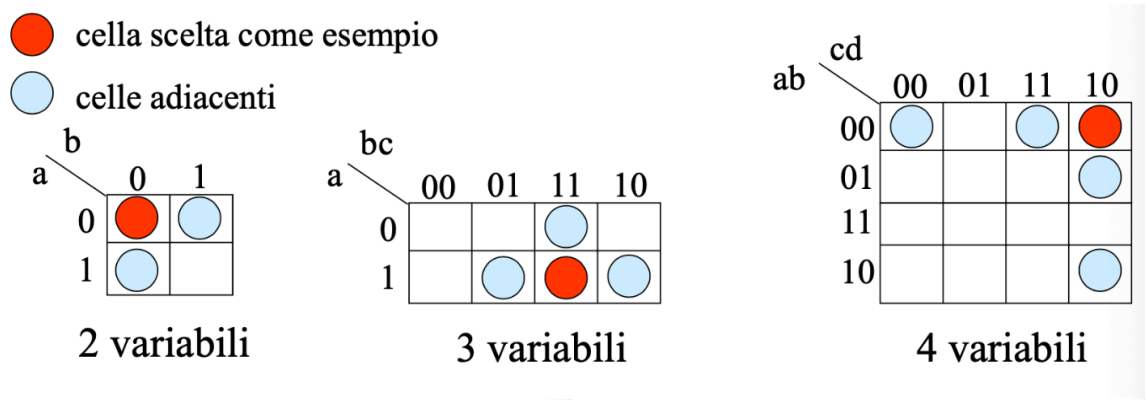
corrispondenza delle diverse combinazioni, quindi qui dentro, a b r andiamo a guardare la prima riga r' 0 ci metto 0 , a 0 b 0 r' ci andiamo a mettere 0 come riporto e così via per tutte.
E ovvio che io non sono costretto per forza a fare prima la tabella di verità è chiaro che uno può fare direttamente la mappa di Karnaugh tanto il principio è lo stesso, posso tranquillamente fare la mappa di Karnaugh e andare a mettere i valori direttamente lì ma se si preferisce si può fare la griglia.

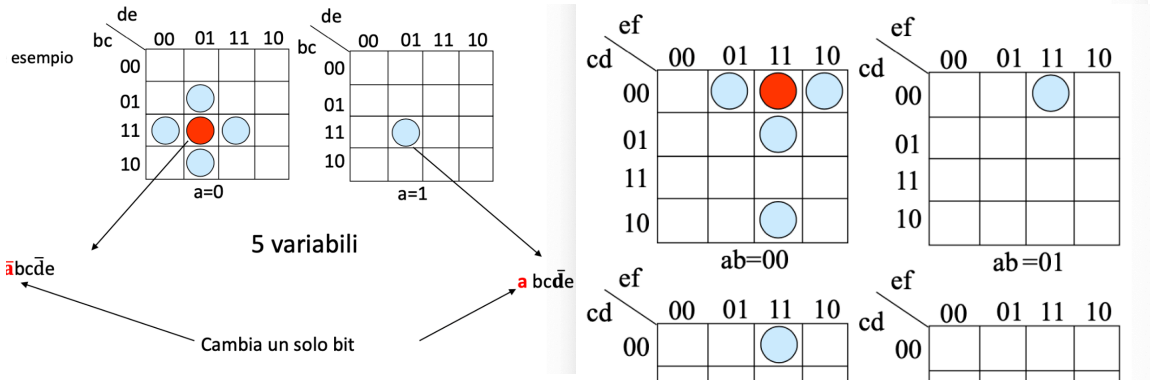
Si ricorda che il bit di parità è un bit che si aggiunge al termine di una sequenza di bit ed è 0, se il numero di 1 è pari, o uguale a 1, se il numero di 1 è dispari (es. 011011→bit di parità=0; 010011 →bit di parità=1).

Nel caso delle tabelle a 4 variabili la mappa torna ad essere nuovamente quadrata e posso andare a mettere due variabili due sulle righe due sulle colonne ed andare a numerare tutte le combinazioni 00-01-11-10, quindi ancora una volta devo stare attento ad avere una distanza di hamming unitaria fra un'etichetta e la successiva, per esempio questa funzione rappresenta è una funzione che calcola la parità su 4 variabili, cos'è la parità?
I bit di parità è un bit che si aggiunge a una sequenza di bit che vale 1 se il numero di bit è pari o vale 0 se viceversa, quindi ad esempio se ho la stringa 0101, 0101 ha un numero di bit pari a 1 pari quindi metto il bit di parità pari a 0 mentre 0111 ha un numero di bit pari a 1 dispari per cui metto il bit di parità pari ad 1, esiste sia la parità dispari sia la parità pari perché io potrei scegliere di mettere 1 se il numero di bit pari a 1 è dispari quindi parità dispari, oppure parità mettendo 1 il bit di parità se il numero di bit che precedono è pari
Quindi all'interno andiamo a mettere solo il bit di parità.
Questa cella qui ci dice che se abbiamo la stringa a b c d che è uguale a 0 0 0 0 tutto 0 il bit di parità sarà 0 perché è un numero nullo di bit pari a 1, se abbiamo 0 0 0 1 il numero di 1 sarà disperati per cui il bit di parità sarà pari a 0, a b pari a 0 0 c d pari ad 1 allora il bit di parità è pari ad 1 e così via, immaginate che questa tabella serve a descrivere come deve funzionare un circuito che ricevendo in ingresso 4 bit ci calcola direttamente il 5 bit che sarebbe il bit di parità.

Adiacenza tra celle

Con la mappa di Karnaugh va introdotto anche il concetto di adiacenza. Due celle sono dette adiacenti se le loro coordinate differiscono per un solo bit.
ln una mappa che descrive una funzione di n variabili ogni cella ha n celle adiacenti. Graficamente, due celle sono adiacenti se hanno un lato in comune o se sono poste all'estremità di una stessa riga o colonna.
È possibile costruire mappe di Karnaugh anche a 5 variabili, utilizzando contemporaneamente due mappe con quattro variabili ciascuna (b, c, d, e) e ponendo in una a=0 e nell'altra a=1. In questo caso è possibile applicare un ulteriore regola di adiacenza: sono adiacenti celle che occupano la stessa posizione in sottomappe adiacenti. Infatti, tra le espressioni abcde e abcde, vi è un solo bit di differenza, rappresentato dalla variabile "a" (le lettere sottolineate sono variabili negate). Infine, vi sono le mappe di Karnaugh a 6 variabili, che utilizzeranno quattro sottomappe, con quattro variabili ciascuna (c, d, e, f), nelle quali ab saranno inizializzate rispettivamente a 00, 01, 10, 11.
Secondo la regola di adiacenza spiegata in precedenza, oltre alle celle adiacenti in una stessa sottomappa, bisogna considerare le celle che occupano la stessa posizione in sottomappe adiacenti, ovvero quelle in cui ab differisce di un solo bit; graficamente, sono adiacenti le sottomappe che condividono almeno un lato.
Le mappe di Karnaugh traducono una regola che è quella della distanza di hamming in un concetto grafico, in modo tale che quando andiamo a fare gli esercizi, anziché stare lì a contare i bit che cambiano andiamo semplicemente a guardare le posizioni, quindi abbiamo di fatto trasformato una regola di distanza matematica in una grafica.





Quando due celle sulla mappa di Karnaugh si dicono adiacenti?
Per cella si intendono questi quadratini che si trovano qua vicino, una cella è adiacente se le rispettive coordinate differiscono per un solo bit, quindi in questa mappa a due variabili, questa cella indicata dal pallino rosso, ha come celle adiacenti le celle che hanno una coordinata cioè il valore di a e b che rispetto ad esse differiscono solo per un valore, siccome questa cella ha come coordinate 0 0 le adiacenti sono quelle che hanno 01 e 10 che sono quelle indicate in celeste.
Se andiamo in una mappa a 3 variabili, prendiamo ad esempio questa cella rossa, quale celle adiacenti ha ?
Le celle che hanno le coordinate che differiscono di un bit, qui di queste qui sarebbe a 1 b1 c1 quindi sarebbe 111 quindi sarà adiacente a 101 che differisce solo di un valore della b, avrà adiacente 011 in cui differisce solo un altro valore, e avrà 110 in cui differisce un altro valore.
Andiamo a questa cella qui, questa mappa a 4 variabili prendiamo come esempio la cella di coordinate 0010, le adiacenti sono per definizione le celle che differiscono per un solo bit quindi sicuramente questi due accanto, perché questo è 0010 questo invece 0011 quindi differisce per 1, ma anche questa differisce per uno anziché il essere 0010 è 0110 ma stranamente anche questa e questa sono adiacenti perche questa è 0010 e questa è 0000 cambia solo 1 come cambia solo 1 anche qui questa è 1010 e quella vicina è 0010. Il motivo per il quale noi nelle mappe di Karnaugh andiamo ad indicare le celle con indirizzi che differiscono solo di 1, è per avere la facilitazione quando le andiamo ad analizzare di identificare come adiacenti fra loro celle che sono adiacenti anche graficamente, cioè il fatto che questa cella ha come adiacenti queste qui è possibile solamente perché abbiamo fatto in modo che si passi da 01 ad 11 e non ad 10, se avessimo rappresentato celle di coordinate che cambiano di 2 bit non sarebbe stato più possibile identificare graficamente come adiacenti le celle celesti rispetto alle rosse, quindi questo è il motivo per cui noi intanto andiamo a mettere i bit che cambiano solo di un valore ma il motivo per cui questa cella qui che risulti adiacenti con queste due è abbastanza intuitivo, ma come fa ad essere adiacente con questa ? (Intendendo quella non direttamente vicina)
Perché sono adiacenti celle che hanno o un lato in comune oppure sono poste all'estremità della stessa riga o della stessa colonna, quindi sono adiacenti le celle che condividono un lato ma anche quando si trovano all'estremità opposta della stessa riga o della stessa colonna e perché lo sono?
Perché se andiamo a guardare gli indici chi si trova all'estremità della stessa riga o della stessa colonna differisce solamente per un bit pur trovandosi agli estremi, questa qui è 0000 e 0010, come quest'altro: 1010 sofferisce solo per un bit da 0010.
Ipotizziamo di lavorare su mappe con 5 variabili quando abbiamo 5 variabili dobbiamo dividere la mappa in 2 sotto-mappe, in sostanza immaginate di avere le 5 variabili a b c d e, purtroppo non posso andare a fare una tabella per calcolare a b da una parte c d e dall'altra perché non riusciremmo a gestire le adiacenze, siamo quindi costretti a fare due mappe ciascuna da 4 variabili, in una mappa andiamo a mettere tutte le combinazioni di valori che assumono ad esempio b - c e d -e quando a è pari a 0, cioè fissò il valore di a pari a 0 e vado ad elencare tutte le combinazioni di b-c sulle righe e d-e sulle colonne, poi vado a fare un'altra mappa in cui vado a mettere a pari ad 1 andando di nuovo ad enumerare tutte le combinazioni di b-c sulle righe e d-e sulle colonne; all'intersezione delle varie celle vado a mettere i valori.
Quindi se io voglio ad esempio specificare qual è il valore che si ha in corrispondenza di a pari a 0 b pari a 0 c pari a 0 d pari 0 ed e pari a 0, vado in questa mappa (la prima in cui si imposta a=0) ad identificare i valori di b-c e d-e e qui dentro ci vado

a mettere o valori della variabile; se voglio mettere invece il valore che si ha quando a è pari ad 1 ed i valori di b c d e sono uguali sa prima devo prenderli da questa mappa e metterli qui.
Regola dell'adiacenza: supponiamo che io ho questa cella qui, la regola dice che sono adiacenti le celle che condividono un lato, ma non solo è adiacente anche questa cella qui, che pur non trovandosi nella stessa mappa si trova in un'altra sotto mappa adiacente a questa cioè accanto a questa che ha la stessa posizione corrispondente, ed in effetti perche questa cella è adiacente a questa perche la regola è che in generale sono adiacenti celle che cambiano solo di un bit, se andiamo a guardare l'unico bit che cambia da qui a qui è proprio la a da una parte vale 0 e da una parte vale 1 ma i valori sia di bc che di de sono sempre gli stessi, ecco perche questa cella è adiacente a questa cambia solo il valore della a.
Quindi accanto alla regola di prima sono adiacenti celle che hanno un lato un comune opposto all'estremità della stessa riga o colonna si aggiunge un'ulteriore regola per la quale sono adiacenti celle che occupano la stessa posizione in sottomappe adiacenti.
Caso più complicato: funzione con 6 variabili, ogni sottomappa ne può contenere massimo 4, avrò 4 sottomappe in cui sostanzialmente vado a scegliere due variabili a e b per etichettare ciascuna sottomappa e ovviamente enumerando tutte le combinazioni, per esempio ab pari 00 ab pari a 01 ab pari ad 10 ed ab pari ad 11, in ogni sottomappa fissando il valore di a e b vado come al solito ad i valori possibili di cdef e faccio 4 mappe quindi quando ad esempio devo andare a scrivere che valore ha la funzione di output quando a è pari a 0 b è pari a 0 c pari a 0 d pari a 0 e pari a 0 ed f pari a 0 lo vado a scrivere qui, quando voglio scrivere quanto vale l output quando a è pari a 1 bcdef pari a 0 lo vado a scrivere qui devo solo guardare l etichette e andare a prendere il punto dove metterle quello che bisogna capire è la regola di adiacenza.
Le sottomappe diventano 4 la prima in alto a sinistra avrà i valori 00 la seconda 01 la terza in basso a sinistra 10 la quarta in basso a destra 11, quando devo andare a calcolare quali sono le celle adiacenti a questa in rosso, sono come sempre secondo la prima regola quelle che sono separate da un bit e sono adiacenti, quella che si trova sul lato opposto della stessa colonna, secondo l'ulteriore regola di adiacenza sono anche adiacenti le celle che si trovano nella stessa posizione delle sottomappe adiacenti (che sono solo 2), una non è adiacente perché deve cambiare solo un bit e quindi mentre da 00 10 01 cambia un bit da 00 a 11 ne cambiano due, quindi tutte le celle evidenziate in celeste sono caratterizzate dal fatto che hanno un indice che differisce solo di un bit dalle coordinate di quella rossa.

Raggruppamento rettangolare

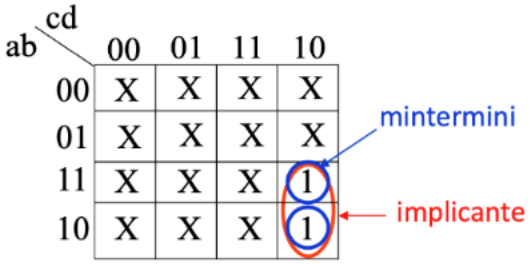
Un raggruppamento rettangolare di ordine p è un insieme di 2p celle di una mappa, all' interno del quale ogni cella ha esattamente p celle adiacenti (es. un raggruppamento rettangolare di ordine 2, avrà 4 celle, ciascuna delle quali sarà adiacente a 2 celle).

Un raggruppamento rettangolare r è sostanzialmente un rettangolo che noi possiamo indicare sulla nostra mappa, che potrebbe essere una singola cella oppure, due celle adiacenti, 4 celle adiacenti, 8 celle adoacenti e così via con un unico requisito stringente che il numero di celle presenti all'interno di questo raggruppamento sia multiplo di due, quindi noi parliamo di raggruppamento rettangolare di ordine p per implicare un insieme di celle, in cui ciascuna delle celle è adiacente a p ciascuna celle. Esempio: in un raggruppamento di ordine 2 sarà un raggruppamento di 4 celle ciascuna delle quali è adiacente alle altre due celle, quindi immaginate 4 celle che fanno un quadrato in realtà potrebbero anche formare un rettangolo.
Con un raggruppamento di ordine 2 si intende 4 celle che potrebbero essere un quadrato o anche una striscia di 4 elementi. Se io ho 4 elementi che formano questo raggruppamento ciascuno di questi raggruppamenti ha due vicini, per questo motivo parliamo di raggruppamento di ordine p, che contiene due alla p elementi ma ciascuno di questi elementi ha p vicini, quindi se io dico raggruppamento di ordine 3 contiene 8 elementi ciascuno di questi avrà 3 vicini secondo la regola di adiacenza.

Raggruppamento rettangolare ed implicanti

Un raggruppamento rettangolare di ordine p ad esempio di ordine 2 o 3, costituito da celle tutte di valore 1, non è altro che un implicante della funzione, un implicante è quel termine che vale 1 dove la funzione vale 1 nella prima forma canonica, quindi tramite un raggruppamento rettangolare di tutti i valori pari a 1 sulla mappa di Karnaugh graficamente possiamo individuare un implicante.
Una volta individuato l'implicante, cioè il raggruppamento rettangolare, quel raggruppamento rettangolare individua un implicante in cui solamente le variabili che assumono valore costante in quel rettangolo compariranno nel risultato, cioè dovremo andare a guardare le etichette sulla mappa, ignoriamo le variabili che hanno i valori che cambiano in quel rettangolo, costruiamo solo le variabili p sempre costante quelle variabili così individuate ci permettono di capire quali variabili considerare nell'output.

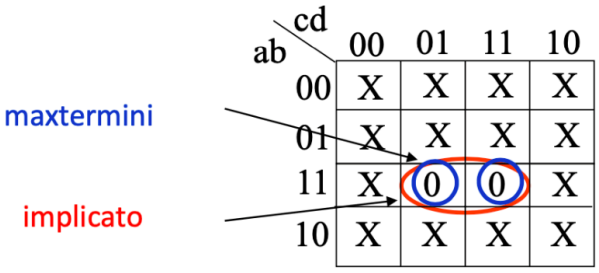
Per ottenere l'espressione semplificata SP (o in forma normale) dell'output, bisogna trascrivere le sole variabili che rimangono costanti nel RR, in forma vera se valgono 1, in forma complementata (ossia negata) se valgono 0 (es. abcd + abcd = acd→la variabile b viene eliminata nell'espressione finale perché assume sia il valore b che b, dimostrandosi non costante). Questa operazione è una velocizzazione di un procedimento più lungo, che sarebbe:
abcd + abcd =
acd (b + b) = una variabile sommata (OR) al suo negato da come risultato sempre 1
acd (1) = una variabile moltiplicata (AND) per 1, da come risultato la variabile stessa.



Esempio questo è una mappa di 4 variabili, in rosso questo raggruppamento rettangolare che è di ordine 1, perché 2 alla 1 fa 2 e quindi queste due celle sono di raggruppamento di ordine 1 perché vuol dire che sono adiacenti.
Questo raggruppamento rettangolare indicato in rosso è quello che corrisponde ad un implicante, perché tutti i termini in questo raggruppamento valgono 1, le x indicano valori indifferenti secondo la definizione in raggruppamento rettangolare contenente tutti 1 corrisponde a quello che è un implicante della funzione somma di valori, cioè corrisponde a quel termine che insieme ad altri termini nella somma raggiunge il risultato finale.
All'interno possiamo individuare due singole celle che sono indicate in blu queste due singole celle corrispondono a quelle che nella forma canonica sd erano chiamati mintermini, cioè quei termini che dopo aver effettuato il prodotto si sommano ad altri mintermini e formano l'output complessivo.
Come faccio a capire i mintermini che valore hanno?
Per capirlo vale la regola che data una cella, quella cella si trasforma in un'espressione che contiene solo le variabili che non cambiano di valore, su una sola cella questo discorso non si può fare, poiché questa singola cella coinvolge le variabili a,b,c e d.
Il meccanismo e lo stesso delle tabelle di verità però se noi identifichiamo un raggruppamento rettangolare che racchiude entrambi i mintermini, possiamo in modo automatico ottenere l'espressione semplificata; e come facciamo a capire che la somma di queste due equivale a questa ?
Di norma dovremmo andare a fare delle elaborazioni con i principi dell'algebra ma deve essere usato questo principio: nel raggruppamento rettangolare Rosso quali sono le variabili che non cambiano di valore la variabile a varia in entrambe 1 quindi non cambia la b da una parte vale 1 da una parte vale 0 quindi va ignorata, c e d valgono sempre 1 e 0 quindi l'espressione corrispondente a quello in rosso comprende sicuramente le variabili a c e d, e dopodiché i valori che prenderemo saranno diretti se la variabile sarà 1 e negata se vale 0 quindi a vale sempre 1 quindi scrivo a, b non la metto perché cambia di valore, c vale 1, d vale 0; quindi si può dimostrare quali siano equivalenti senza fare un calcolo ma è una forma minimizzata perché contiene meno termini e anche meno letterali.
X*y + x*y negato è uguale ad x in realtà come si fa x si mette in evidenza e quindi hai x che moltiplica y or y negato che vale sempre 1 quindi sarebbe x and 1 = x.
Sono due le proprietà che si usano questa è la distributiva al contrario e quella del complemento, cioè che a or a negato in questo caso b o b negato è uguale ad 1 e poi la proprietà dell'identità per cui x and 1= x, passare da questa espressione a questa facendo due elaborazioni banalissime è facile ma farlo graficamente è ancora più facile.
Una volta che identifico un raggruppamento di tutti valori 1, l'espressione corrispondenti a quel raggruppamento si calcola prima, andando a discutere le variabili, che in quel raggruppamento cambiano di valore quindi che non hanno un valore costante, poi andando ad usare la regola del: valore 1 variabile diretta valore 0 variabile negata.

Raggruppamento rettangolare ed implicati

Un RR di ordine p costituito da celle contenenti tutte valore 0, ed eventualmente condizioni di indifferenza, individua un implicato della funzione. In rosso è indicato l'implicato ed in blu i maxtermini.
Per ottenere l'espressione semplificata PS (o in forma normale) dell'output, bisogna trascrivere le sole variabili che rimangono costanti nel RR, in forma vera se valgono 0, in forma complementata se valgono 1 (es. abcd + abcd = abd→la variabile c viene eliminata nell'espressione finale perché assume sia il valore c che c, dimostrandosi non costante).



Un raggruppamento rettangolare avente valori tutti 0 rappresenta di fatto un implicato della funzione in forma canonica prodotti di somme e poi le variabile devono essere prese dirette se valgono 0 negate se valgono 1, cioè il contrario di prima.
Supponiamo che io ho una mappa di Karnaugh e anziché voler ragionare in prima forma canonica voglio lavorare in seconda forma canonica cioè in prodotti di somme non somme di prodotti, basta ragionare al contrario cioè vado ad identificare i raggruppamenti rettangolari aventi valori 0 ad esempio questo due quindi qui abbiamo un raggruppamento di ordine 1 con due delle ciascuna delle quali adiacenti all'altra questa cella continente 2 oggettini piccoli questi oggettini piccoli di valore 0 non sono più mintermini ma sono Maxtermini e l oggettino rosso non è più un implicante ma un implicato.
Se ho questi due maxtermini nella seconda forma canonica dobbiamo moltiplicarli tra loro quindi dobbiamo fare questo termine che è una somma, questo termine che è una somma ed entrambi li moltiplichiamo e poi come andiamo a mettere qui i valori di a, b, c, d al contrario di come facciamo nell'altra forma canonica quindi qua ad esempio a vale 1 quindi la mettiamo negata b =1 la mettiamo negata, c vale 0 la mettiamo diretta d vale 1 la mettiamo negata e la stessa cosa qui a negato b negato c negato perché sono tutti pari ad 1 questa qui è l'espressione canonica ps relativa a questa qui.
Io non ho bisogno di andare a scrivere questi e poi calcolare posso farlo direttamente andando ad individuare il raggruppamento rettangolare in rosso di dimensione 2 ovvero di ordine 1 e andando a seguire la regola, cioè questo raggruppamento rettangolare contenente tutti i valori 0 corrisponde ad un implicato della forma ovviamente ps contenente solo le variabili che non cambiano di valori in questo caso sono a, b, d perché la c cambia e poi dobbiamo andarle a prendere in formato negato se valgono 1 in formato diretto se valgono 0 questa è l'espressione ps cioè prodotti di somma in cui dobbiamo fare a negato or b negato or d negato.
La prossima volta dovremmo cercare di individuare tutti i raggruppamenti possiamo decidere di lavorare in forma ps o sp e poi se lavoriamo in forma ps andiamo ad individuare raggruppamenti che individuano tutti gli 0 che esistono sulla mappa una volta individuati tutti andiamo a tradurli nelle corrispondenti formule li andiamo a moltiplicare e quella sarà l'espressione minima corrispondente alla tabella.

Quindi è un lavoro prettamente meccanico con Sforzo intellettuale —> nullo.

APPROFONDIMENTO

Mintermini e maxtermini

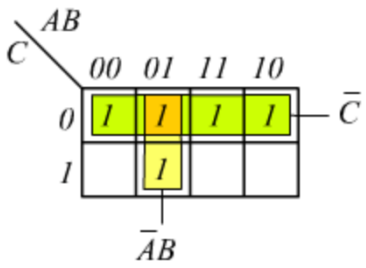
È definito mintermine, (termine minimo indicato con m) il prodotto in cui ogni variabile logica di ingresso appare una sola volta, complementata (negata) o non complementata a secondo che essa appaia nella tabella della verità della funzione con valore 0 o con valore 1.
Si definisce maxtermine, (termine massimo indicato con M) una somma in cui ogni singola variabile della funzione è presente complementata o non complementata a seconda che appaia nella tabella della verità con il valore 1 o con il valore 0.
La somma dei mintermini delle combinazioni che determinano un livello 1 della funzione logica costituisce la forma canonica della somma. Il prodotto dei maxtermini delle combinazioni che determinano un livello 0 della funzione logica,costituisce la forma canonica del prodotto di una assegnata tabella della verità.
Come si verifica facilmente, data la tabella della verità, la funzione logica associata può essere espressa tramite una di queste due forme.
La forma canonica della somma o del prodotto non è necessariamente l'espressione più semplice adatta a rappresentare una data funzione booleana.
Una funzione minimizzata comporta un circuito logico più semplice e di conseguenza a più basso costo ed ingombro rispetto a quello realizzato partendo dalla funzione non minimizzata.

Mappa di Karnaugh

La tabella della verità è una sintesi del comportamento della variabile di uscita Y in corrispondenza di tutti i possibili valori delle variabili di ingresso. Dalla tabella della verità di una funzione si può subito ricavare la corrispondente mappa di Karnaugh.
La mappa di Karnaugh di una funzione ad n variabili di ingresso consiste in un rettangolo di 2n caselle, dove ogni casella corrisponde ad uno dei possibili stati (combinazioni) delle variabili di ingresso con la caratteristica che passando da una casella all'altra in ogni direzione (ma non in diagonale) cambia una sola delle variabili. Debbono essere considerate adiacenti anche le caselle di estremità di una riga o di una colonna, come se la mappa fosse disegnata su una superficie chiusa su se stessa. In ognuna delle caselle viene posto il valore assunto dalla funzione (variabile Y) booleana per quello stato (per quella combinazione delle variabili di ingresso).
Le regole che si seguono per la semplificazione sono:

1. Bisogna individuare il minor numero di gruppi (che copre tutti gli 1 della mappa).
2. Ciascun gruppo deve contenere il maggior numero di 1 adiacenti (il numero di 1 che costituisce un gruppo deve formare una potenza del 2, si scelgono perciò gruppi da due 1 o da quattro 1 etc..).
3. Come si è detto sono considerabili adiacenti le caselle di estremità.
4. Eventuali 1 isolati costituiscono un gruppo e debbono essere riportati integralmente.

5. Da ogni gruppo si estrae un termine che contiene le variabili di ingresso che non variano passando da una casella all'altra del raggruppamento stesso, ciascuna variabile sarà in forma vera o negata a secondo se vale 1 o 0 nel raggruppamento.
6. La funzione logica minimizzata sarà data dalla somma logica dei termini estratti dalla mappa.



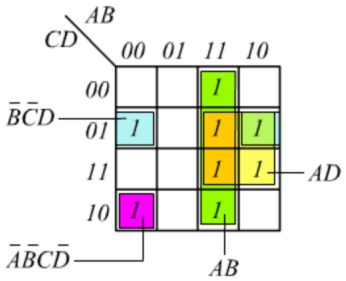
Come si nota il minimo numero di gruppi è due, ciascun gruppo contiene tanti 1 sempre in ragione delle potenze del 2 (cioè 4 e 2 uno) nel primo gruppo orizzontale passando da una casella all'altra l 'unica variabile che non cambia è la C che rimane impostata a 0 per cui verrà estratta come termine C.

Nel gruppo verticale passando da una casella all'altra le nice variabili che non cambiano sono la A che rimane impostata a 0 (A) e la B che rimane impostata a 1 (B); verrà dunque estratto il termine AB; in definitiva la funzione semplificata sarà:

$$Y = \overline{A}B + \overline{C}$$

Supponiamo ora di dover semplificare una funzione con 4 variabili, la rappresentazione usuale della mappa sarà la seguente:

$$Y = \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}C\overline{D} + AB\overline{C}\overline{D} + AB\overline{C}D + ABCD + ABC\overline{D} + A\overline{B}CD + A\overline{B}\overline{C}D$$
$$0001 \quad 0010 \quad 1100 \quad 1101 \quad 1111 \quad 1110 \quad 1011 \quad 1001$$



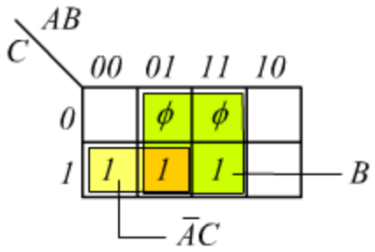
In questo caso si evidenziano le due celle non contigue (disgiunte di estremità) sulla seconda riga da cui viene estratto il termine
B C D e il termine isolato in basso a sinistra che deve essere riportato integralmente.
Nella colonna verticale passando da una casella all'altra non cambiano le variabili AB che rimangono impostate a 1 (AB) mentre il gruppo da 4 quadrato non vede cambiare le Variabili A che rimane impostata a 1 e D che ugualmente rimane a 1 (AD).
La funzione risultante sarà:

$$Y = AB + AD + \overline{B}\overline{C}D + \overline{A}\overline{B}C\overline{D}$$

Condizioni di indifferenza

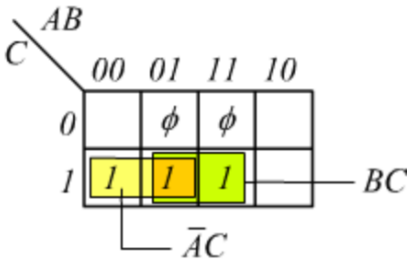
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	ϕ
0	1	1	1
1	0	0	ϕ
1	0	1	0
1	1	0	ϕ
1	1	1	1

Talvolta il valore dell'uscita in una data tabella della verità non viene specificato per alcune combinazioni delle variabili di ingresso o perchè queste combinazioni non possono verificarsi o perchè non interessa conoscere il valore della variabile di uscita corrispondente a tale combinazione. Si parla in questo caso di condizioni di indifferenza. In questa situazione l'uscita che può assumere indifferentemente il valore 0 o 1 viene riportata sulla mappa e sulla tabella col simbolo ϕ (talvolta si usa anche il simbolo X). Le condizioni di indifferenza possono essere sfruttate per semplificare ulteriormente la funzione assegnando loro il valore 1 quando ciò risulti conveniente.



In questo caso considerando 1 le condizioni di indifferenza abbiamo:

Y=(!A)C+B



Nel caso avessimo considerato 0 le condizioni di indifferenza la funzione risultante sarebbe stata leggermente più complessa:

Y=(!A)C+BC

Concetto di adiacenza

Concetto di adiacenza geometrica: si intende una relazione che lega le cifre immediatamente vicine orizzontalmente e verticalmente, tenendo anche conto della cilindricità orizzontale e verticale della mappa di Karnaugh, che rende adiacenti tra loro la cifra di un estremo con quella dell'estremo opposto.
Concetto di adiacenza logica: si intende una relazione che lega le cifre le cui coordinate differiscano per un solo valore; tale differenza si definisce "distanza di Hamming" e dovrà essere unitaria.
I riferimenti orizzontali e verticali della matrice vanno dunque disposti nell'ordine 00 01 11 10 affinché l'adiacenza geometrica tra gli elementi della matrice coincida con l'adiacenza logica.