

ARCHITETTURE DI CALCOLO LEZIONE 7

Reti combinatorie

Per ricapitolare:

Sostanzialmente in una mappa di karnaugh possiamo trovare dei raggruppamenti rettangolari (si intende anche quadrati ma comunque con una numerosità pari ad una potenza di due).

Devono quindi essere dei raggruppamenti caratterizzati da valori tutti pari a 1 o tutti pari a 0, è ammesso raggruppare anche dei valori indefiniti (x o - che rappresentano i valori DON'T CARE), questi raggruppamenti identificano degli implicati o degli implicanti quindi dei valori implicati nelle formule del circuito.

Un raggruppamento è un insieme di 2^p celle, dove p è l'ordine del raggruppamento, contenente solo 1 e valori indifferenti (forma SP) o solo 0 e valori indifferenti (forma PS).

I raggruppamenti individuano ciascuno un implicante (forma SP) o un implicato (forma PS), rispettivamente contenenti mintermini e maxtermini, e ci permettono di trascrivere la forma normale, composta da tutte le variabili che rimangono costanti; saranno in forma vera se uguali a 1 e in forma negata se uguali a 0 quando operiamo nella prima forma canonica (SP), oppure in forma vera se uguali a 0 e in forma negata se uguali ad 1 quando operiamo nella seconda forma canonica (PS).

È possibile calcolare quante sono le variabili costanti per ogni raggruppamento grazie alla formula $n-p$, dove n è il numero di variabili e p è l'ordine del raggruppamento.

Facendo riferimento alla figura, $n=4$ (a, b, c, d) e $p=2$ ($2^2=4$ celle), per cui il numero di variabili costanti è 2 a bd.

	cd	00	01	11	10
ab	00	X	X	X	X
	01	X	1	1	X
	11	X	1	1	X
	10	X	X	X	X

Implicante/implicato primo

Implicante primo

Un RR formato da celle contenenti valore "1" o "-" (valore indifferente) e non interamente incluso in un RR di ordine superiore individua un implicante

	cd	00	01	11	10	
ab	00	X	1	1	X	
	01	X	1	1	X	
	11	X	1	1	X	
	10	0	1	1	X	

bd non è un implicante primo !
d è un implicante primo !

primo. Bisogna quindi scrivere tutte le variabili che compaiono in questa cella, in formato diretto se valgono 1 o in formato negato se valgono 0, quando abbiamo più mintermini adiacenti possiamo raggrupparli in un implicante che avrà una formula la quale attraverso la messa in evidenza o anche senza ricorrere all'algebra attraverso un metodo automatico può essere ridotta, questo procedimento necessita di trovare dove si trovano le variazioni di valore degli input, quelle che non variano e valgono 1 vanno considerate in forma diretta mentre quelle che valgono 0 in forma negata mentre quella che variano non devono essere considerate.

Implicato primo

Un RR formato da celle contenenti valore "0" o "-" (valore indifferente) e non interamente incluso in un RR di ordine superiore individua un implicato primo. Quando andiamo a ragionare con la forma canonica di secondo tipo quindi prodotti di somme, il ragionamento è pressoché lo stesso, solo che andiamo a individuare i raggruppamenti rettangolari di 0 e questi raggruppamenti prendono il nome di implicati e non implicanti, si individuano sempre i valori fissi solo che questa volta se prendono in maniera diretta i valori di input pari a 0 e in forma negata i valori pari a 1. Anche qui c'è un modo di ridurre le forme in modo automatico bypassando le formule algebriche, raggruppiamo non i maxtermini questa volta ma i raggruppamenti più grandi di 0 e andiamo a prendere gli input che in questo raggruppamento non variano e poi scriviamo in forma negata i valori di input che valgono 1 e in maniera diretta quelli che valgono 0.

Qui c'è un esempio in cui riusciamo a trovare sulla mappa quello che è un raggruppamento rettangolare di dimensione 4, ovvero di ordine 2^2 , contenente quattro celle di cui ognuna è vicina ad altre 2, andiamo quindi a trovare la formula di ciascuno di questi mintermini, questa espressione formata dalla somma di tutte le formule delle celle contenenti i mintermini, può essere semplificata sia con l'algebra e quindi raggruppando i valori da questa espressione in forma estesa, oppure si può ridurre applicando la regola di cui abbiamo parlato prima, ovvero esploriamo quei termini che non variano ovviamente in forma negata o diretta in base alla forma che stiamo utilizzando (prima o seconda forma canonica).

Nel prodotto risultante compaiono solo le n-p variabili che rimangono costanti nel calcolo del raggruppamento, dove n è il numero delle variabili mentre p è l'ordine del rettangolo. Come si nota dall'immagine, con un n costante, più aumenta p e più diminuisce il numero di variabili costanti; infatti, nel raggruppamento rosso ne avremo 2 ($4-2=2 \longrightarrow b, d$) mentre nel raggruppamento blu solo 1 ($4-3=1 \longrightarrow d$), ottenendo così un'espressione minima finale più sintetica. Nel caso in cui il numero di varianti sia uguale all'ordine del raggruppamento, le variabili saranno tutte costanti e tutti i

	cd	00	01	11	10
ab	00	1	1	1	1
	01	1	1	1	1
	11	-	1	-	1
	10	1	1	-	1

	cd	00	01	11	10
ab	00	0	1	x	0
	01	0	x	x	0
	11	0	x	x	0
	10	0	x	1	0

valori contenuti nella mappa di Karnaugh risulteranno uguali, ottenendo un output indipendente dalle variabili.

Se ho una mappa che ha come risultati tutti 1, il risultato di output sarà 1 e sarà costante e non dipenderà quindi dalle variabili di input, ma questo è un caso speciale, per cui n-p sarà 0 perché t racchiude tutto.

È intuitivo capire che maggiore è la dimensione del raggruppamento rettangolare, minore sarà il numero delle variabili coinvolte, fino al caso estremo in cui il raggruppamento sarà di dimensione 1 quindi ordine 0 in cui le variabili coinvolte saranno n, ma così si tornerebbe al caso dei mintermini che troviamo già nell'espressione canonica per cui non ci sarebbe alcuna semplificazione.

Qui abbiamo due implicant uno in rosso e l'altro in blu, questo in rosso però non è un implicante primo poiché può essere racchiuso in un implicante più grande, quindi possiamo dire che un implicante primo è un raggruppamento rettangolare non completamente incluso in un raggruppamento rettangolare di ordine superiore, se un raggruppamento rettangolare non è incluso in uno più grande allora sarà un implicante primo poiché verrà incluso nella espressione della tabella, ovviamente lo stesso concetto esiste e viene riportato con l'implicato primo.

Passando da implicant più piccoli ad uno più grande che può racchiuderli ovviamente l'espressione diventa più piccola e viene quindi minimizzata.

Avendo due forme canoniche, quando si realizzano i raggruppamenti, è bene scegliere la forma SP, nel caso in cui vi sia una prevalenza di 1, o la forma PS quando vi è un'abbondanza di 0, per giungere a formule più sintetiche.

Per ottenere implicant/implicant primi vi è bisogno che il raggruppamento sia massimo.

I due singoli raggruppamenti (in rosso) non rispettano questo principio, in quanto possono essere considerati un unico insieme di celle per via della regola dell'adiacenza; infatti, ogni cella con c=0, d=0 è all'estremità opposta della mappa di Karnaugh rispetto alle celle con c=1, d=0.

La formula corretta di questo raggruppamento, quindi, è "d" e non "c+d" (forma PS). Bisogna fare attenzione, quando si calcola la forma minima, a non considerare gli RR le cui celle sono tutte incluse in altri RR. Infatti, la forma normale SP della mappa di fianco sarebbe " $acd + abc + abd$ ", che ridotta, seguendo le regole algebriche, è " $acd + abd$ ", ovvero la forma che avremmo ottenuto considerando solo i raggruppamenti blu e verde (il raggruppamento rosso è ridondante).

L'obiettivo finale è quello di individuare la copertura minima, ovvero un insieme di raggruppamenti rettangolari che non ha ridondanze e che racchiude tutte le celle con valore 0 (forma PS) o tutte le celle con valore 1 (forma SP).

Quindi, individuando una copertura con il minor numero di RR di dimensione massima, si deriva un'espressione minima della funzione.

		cd			
		00	01	11	10
ab	00	X	X	X	X
	01	X	X	X	X
	11	X	1	1	1
	10	X	X	X	1

Vediamo qualche altra considerazione, noi possiamo avere un raggruppamento rettangolare che si sovrappone a uno o più raggruppamenti rettangolari adiacenti, questo raggruppamento in rosso che comprende due celle adiacenti si interseca con altri due raggruppamenti, questo in blu e questo in verde, e si interseca in modo tale che tutti gli elementi che esso contiene vengono racchiusi negli altri raggruppamenti per cui questo raggruppamento in realtà risulta ridondante, e noi non dobbiamo prendere ovviamente dei raggruppamenti che

vengono già racchiusi in altri poiché così facendo creeremo delle espressioni più complesse che sono equivalenti ad espressioni più brevi e risulterebbero come già detto prima ridondanti quindi inutili.

In sostanza quelli che dobbiamo fare è cercare tutti i raggruppamenti più grandi che racchiudono tutti gli uni presenti e stare attenti a non racchiudere raggruppamenti inutili, questo discorso vale per tutti e due i tipi di formule canoniche.

Quindi l'espressione equivalente a questi tre raggruppamenti o implicati, sono queste tre messe in and (moltiplicate) tra loro, e usando delle proprietà ci accorgiamo che possono essere ridotte ma ciò è possibile già intuirlo a livello grafico vedendo che uno dei raggruppamenti viene incluso dagli altri.

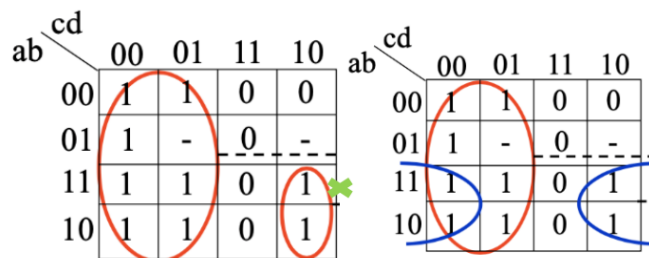
Esempi

- Il RR (x) della mappa di sinistra non è minimo, in quanto dovrebbe includere anche le celle "1100" e "1000", che risultano adiacenti alle celle "1110" e "1010"; ne deriva che l'espressione non è a sua volta minima ($c + acd$). Nella mappa di destra (versione corretta) si ha una copertura minima e l'espressione ottenuta è $c + ad$.
Nota: il RR (x) non può andare ad includere anche le celle "1101" e "1001" perché ogni raggruppamento rettangolare contiene 2^p celle, e 6 non può essere ottenuto da questa formula.
Condizione di indifferenza: l'output non si presenta mai oppure l'output generato non ci interessa.

Adesso vi farò una serie di esempi completi in cui capiremo l'importanza di trovare il minor numero di raggruppamenti di dimensione massima e cosa accade se non li troviamo, prendiamo il caso di sopra, è una mappa di karnaugh a 4 variabili, ricordiamo che questi trattini sono dei valori di indifferenza (don't care ovvero valori di cui non ci interessa l'output) e nel caso li includessimo con gli 1 non stiamo altro dicendo che stiamo formando un'espressione in cui quegli output valgono 1, ma a noi non interessa in quel caso l'output che valore assume.

Nel caso anche la tabella di verità abbia la presenza di trattini ed in quel caso si parla di funzioni non completamente specificate, ovvero stiamo dicendo che la funzione non ci dice quali sono tutti i valori possibili di output, per gli input valutati, ma solo di un determinato sottoinsieme.

Ipotizziamo di aver racchiuso questi due raggruppamenti, questi due raggruppamenti definiscono una copertura minima? Il primo è sicuramente un raggruppamento di dimensione massima, il secondo invece non è un raggruppamento massimo poiché ci siamo dimenticati che per via delle adiacenze potevamo raggruppare nell'espressione questi altri due valori, ma non nel raggruppamento rettangolare perché i valori sarebbero stati 6 e il raggruppamento può ricevere solo un numero di valori pari a 2^n , analizzando la mappa ci rendiamo conto però che le forme equivalenti sono: 1^a) c negato (a,b,d variano mentre la c è fissa ma vale 0), 2^a) $ac(d$ negata), se estendiamo queste espressioni per comprendere anche gli altri due valori la rossa vale sempre c negato mentre questa blu perde la variabile c quindi la forma totale prenderà a diretto e d negato (Si può semplificare perché è sarebbe stato un circuito and a 3 ingressi di cui uno valeva sempre 1 per cui si può semplificare semplicemente trasformandolo in un circuito a due ingressi); quello in blu viene conservato perché solo parzialmente sovrapposto quindi è necessario perché copre degli 1 che altrimenti non potrebbero essere inclusi.



- In questo caso vi è l'esempio di un RR ridondante; nella mappa di sinistra vi sono, infatti, quattro raggruppamenti che comportano un'espressione di quattro termini ($\underline{acd} + \underline{abc} + \underline{bcd} + \underline{ac}$). Nella mappa di destra, invece, ci sono solo tre RR, realizzando così una copertura minima e un'espressione con un minor numero di termini ($\underline{acd} + \underline{abd} + \underline{ac}$).

Questa espressione però non è un'espressione minima poiché ci sono sovrapposizioni tra i raggruppamenti e questi ultimi sono il più grandi possibili, ma purtroppo il numero di rettangoli non è il più piccolo possibile, questo perché noi abbiamo 4 termini ma è facile verificare che si può ottenere lo stesso risultato con 3 termini ... come?

		cd			
		00	01	11	10
ab	00	0	0	0	1
	01	0	1	1	1
	11	1	1	0	0
	10	1	1	0	0

		cd			
		00	01	11	10
ab	00	0	0	0	1
	01	0	1	1	1
	11	1	1	0	0
	10	1	1	0	0

Non consideriamo questi due rettangoli uno e due, li eliminiamo e ne abbiamo qui un altro come facciamo qua, perché abbiamo comunque rispettato la regola che sono racchiusi tutti gli 1, non ci sono rettangoli di dimensioni minima ma sono tutti di dimensione massima rispetto alla zona in cui si trovano, ma il numero di rettangolo scende da quattro a tre, quindi l'espressione si minimizza ricordiamoci che ogni volta che cacciamo un termine stiamo rimuovendo una porta logica dal nostro circuito.

Quindi quell'espressione ovviamente questo rettangolo di rimane sempre $a \cdot (c \text{ negato})$ questo è $(a \text{ negato}) \cdot c \cdot (d \text{ negato})$ per questa espressione che sopra non c'era ma comunque è facile da calcolare, $(a \text{ negato}) \cdot b \cdot d$ la C non c'è perché cambia la d attiva perché va da uno a uno, questa è l'espressione minima associata alla mappa di karnaugh che descrive una funzione assegnata.

- Infine, esaminiamo un caso di espressioni minime PS equivalenti. I raggruppamenti sulle due mappe hanno la stessa complessità, infatti, il numero di termini è lo stesso, così come lo è il numero di letterali per termine.

[a sinistra: $(a+b+c)(a+b+d)(a+b+c)(a+b+d)$; a destra $(b+c+d)(a+c+d)(b+c+d)(a+c+d)$]

Quello che abbiamo visto prima lo si può usare anche per lavorare sulle mappe non costruite ricoprendo di uno ma ricoprendo gli zeri, quindi se lo vogliamo con le mappe non cambia il ragionamento, devo sempre trovare le più grandi i più grandi raggruppamenti di zeri e che siano il minor numero possibile.

		cd			
ab		00	01	11	10
		00	01	11	10
00	1	0	0	1	
01	1	1	0	0	
11	0	1	1	0	
10	0	0	1	1	

		cd			
ab		00	01	11	10
		00	01	11	10
00	1	0	0	1	
01	1	1	0	0	
11	0	1	1	0	
10	0	0	1	1	

Vediamo un esempio allora io mi ritrovo questa mappa e la prima cosa che dobbiamo fare come vedremo e decidere se partire dagli 0 o dagli 1, quindi se voglio fare una forma ps o sp, diciamo

che voglio fare una ps ovvero una forma prodotti di somme, allora creo questo raggruppamento qui, poi questo qui è un altro raggruppamento convenite con me immagino di sì, perché questa cella è adiacente a questa qui poiché si trovano agli estremi della stessa colonna, quindi questo è un raggruppamento, poi questo è un'altra raggruppamento e infine questo è un raggruppamento; diciamo secondo voi si può fare di meno di quello che c'è qui?

Riuscite ad individuare la possibilità di costruire raggruppamenti più grandi di dimensione a base due?

Visto che la risposta è no possiamo passare alla fase finale ovvero scrivere l'espressione, si fa il solito modo, per ciascun raggruppamento che essendo di dimensione massima si chiamano implicati primi, quindi questi sono implicati primi, e come si passa da un implicato primo alla corrispondente formula?

Dividiamo sempre le variabili quelle che non variano e quelle che variano, poi l'unica cosa è che le prendiamo dirette se valgono 0 e negare se valgono 1; poi le variabili non le moltiplichiamo fra loro ma le sommiamo.

La and equivale ad andare a fare per mentre la or equivale a sommare.

Chi ha fatto questa mappa di karnaugh, che quindi ha fatto i raggruppamenti con la copertura minima ha fatto una scelta, ma c'è un'altra scelta altrettanto valida, che è mostrata sotto, cioè potevamo raggruppare questi due, questi due, questo con questo e questi due, giusto e questa è un'alternativa che produce un'espressione ovviamente diversa ma equivalente che ha la stessa complessità cioè lo stesso numero di termini o lo stesso numero di variabili e di conseguenza sono due espressioni altrettanto valide, che derivano semplicemente da un diverso modo di individuare le coperture cioè di raggruppamenti sulle mappe.

Possono esistere più espressioni, ovviamente se ne trovi una che è più semplice delle altre, vuol dire che una è minima mentre l'altra no. Può capitare che io costruisca due tipi di raggruppamenti uni con gli uni e uno con gli zeri e che ottenga la stessa identica espressione.

Queste cose non servono solo a progettare circuiti combinatori ma anche per progettare i circuiti sequenziali nei quali si usano gli stessi principi, quindi queste cose si risolvono con semplici automatismi.

Ricordiamoci che la regola grafica non è altro che un'implementazione della regola logica, ovvero sono considerate adiacenti due celle che differiscono solo per un bit.

Casi particolari

- Il RR coincide con l'intera mappa di Karnaugh

Lo abbiamo visto prima ma lo spiego praticamente, una cosa del genere ci dice tutti questi valori sono 1, questi sono valori di indifferenza quindi se ragioni in termini di uno come ovvio, non ci sono zeri, questi valori di indifferenza li possono racchiudere e viene il rettangolo più grande possibile di dimensione 16, in cui ovviamente stiamo dicendo che questa funzione darà sempre uno indipendentemente dai valori delle variabili, quindi l'output è identicamente uguale a loro, cioè sostanzialmente è un caso un po' speciale.

		cd			
		00	01	11	10
ab	00	1	1	1	1
	01	1	1	1	1
	11	-	1	-	1
	10	1	1	-	1

- L'espressione minima coincide con l'espressione canonica

Guardate questa funzione, praticamente se andiamo a per edificare qui i raggruppamenti e la copertura minima chiaramente sono questi, questi singoli termini, questi sono rettangoli di dimensione uno e abbiamo visto che un rettangolo di dimensioni uno comprende tutte le variabili, quindi a, b, c e d, in effetti se andiamo a scrivere l'espressione equivalente, questi termini sono esattamente i termini dell'espressione canonica, cioè se noi andiamo a scriverlo tabella di verità di questo come un vecchio modo cioè scriviamo tutti gli 8 casi e devo individuare le tre righe in cui ci sono i vari uno, e andiamo a sommare i termini prendendoli diretti e negati ci esce questa espressione qui, cioè in sostanza in questo caso la mappa di Karnaugh non ci servirà a nulla perché ci ha tirato fuori l'espressione minima che esattamente l'espressione canonica in prima forma normale, cioè in altri termini ci sono dei casi ancora rari come questi in cui non riusciamo a minimizzare l'espressione, perché l'espressione già calcolata sulla tabella di verità nel modo che conoscete è già finita.

Non è neanche difficile intuire quando è che capita, capita quando sostanzialmente i valori sono alternati, non riusciamo mai a creare raggruppamenti perché non capita mai che ci sono 2 valori 1.

		bc			
		00	01	11	10
a	0	0	1	0	0
	1	1	0	1	0

- Le coperture minime PS e SP portano alla stessa espressione (PS: un solo termine composto da due variabili; SP: due termini composti ciascuno da una sola variabile)

Questo invece è un altro caso tanto simpatico diciamo così in cui noi possiamo decidere di ragionare in termini di prima forma normale o seconda forma normale e ottenete esattamente lo stesso risultato, vediamo perché allora vedete si tratta di una mappa in cui ci sono solo quattro zeri ai bordi e agli angoli tutto il resto è uno, ci sono un paio qui di valori trascurabili proviamo a ragionare prima con la forma somma di prodotti, che sembra anche ragionevole perché ci sono tanti uno, di solito si sceglie di usare in uno se sono di più di uno e zero se sono di più di zero, quindi

ab \ cd	00	01	11	10
	00	0	1	1
01	1	1	1	-
11	1	1	1	1
10	0	1	1	0

PS: $b + d$

SP: $b + d$

PS: $b + d$

SP: $b + d$

cominciamo qui ed individuiamo questo raggruppamento e quest'altro, notate che stiamo includendo anche questi trattini ed è giusto che sia così, cominciamo con le espressioni equivalenti, quindi facciamo la somma di prodotti, allora in questo rettangolone qua dobbiamo vedere le quattro variabili quali sono le variabili che non variano, la d è l'unica fissa che non cambia, di questo invece rettangolo vale B perché vale sempre uno e gli altri cambiano, quindi questi due termini sono i due termini prodotto, dobbiamo sommare questi due termini e teniamo $B + D$ e questa è la somma di prodotti.

Facciamo adesso il ragionamento sugli zeri prodotti risolvere allora ci sono quattro zeri agli spigoli, questi formano un raggruppamento di dimensione quattro, perché sono tutti e quattro vicini, questa matrice va immaginata come una specie di cilindro, immaginando che si gira su se stessa e gli estremi si toccano e quindi abbiamo questo singolo raggruppamento, qual è l'espressione equivalente PS ?

Consideriamo le variabili che cambiano e quelle che non cambiano, in questo caso è un unico termine perché è un unico raggruppamento; che sia chiaro questi qui erano due termini sommati fra loro ma erano due termini di ciascuno di una variabile, questo invece un unico termine di due variabili che vanno ad essere sommate fra loro, questo perché nella seconda forma si usa la somma e le due variabili che qui non variano sono b e d, perché la a varia, la c varia pure qui vale zero e qui vale uno, mentre la d vale sempre 0 come la b vale sempre zero, quindi b e d le dobbiamo prendere dirette perché valgono zero, l'esatto contrario dell'altra situazione quindi stiamo sommando b e d; abbiamo ottenuto esattamente la stessa espressione minima usando sia la copertura sugli zeri che la copertura su uno, è un caso un po particolare ma possibile.

L'obiettivo è quello di ottenere un insieme di raggruppamenti, ovvero ciò che in gergo si chiama COPERTURA MINIMA, che sia costituita dal minor numero possibile di raggruppamenti rettangolari ovvero senza ridondanze.

La copertura è un insieme di raggruppamenti rettangolari la cui unione racchiude tutte le celle con valore 1 nella prima forma canonica o di tutti 0 se si tratta della seconda, le coperture che noi andiamo a costruire sulla mappa determinano le espressioni che si derivano da quest'ultima, quindi se andiamo a trovare il minor numero di raggruppamenti ma con la massima dimensione quindi determiniamo la copertura minima automaticamente determineremo quindi quello che è l'espressione minima della mappa.

Ci vuole un minimo di fantasia per risolvere le mappe di Karnaugh per fortuna minima, in altri termini non possiamo dare una regola assoluta per calcolare le mappe di Karnaugh ma possiamo dare delle linee guida, le quali portano sempre comunque con un minimo di esercizio a risolvere il problema; c'è una serie di passi che dobbiamo seguire.

Fasi per l'individuazione grafica dell'espressione minima

1. Si ricava la mappa di Karnaugh dalla tavola di verità.
2. Si decide se lavorare con la forma SP o PS.
3. Si cerca di individuare tra le celle da coprire una cella che possa essere racchiusa in un solo RR e lo si traccia di dimensione massima, annotando il termine corrispondente. Se la funzione è incompleta, il RR può contenere anche condizioni di indifferenza.
4. Si ripete fino a quando è possibile il passo 3, tenendo conto della possibilità di coprire anche celle incluse in RR già tracciati.
5. Si prendono in considerazione le celle ancora da coprire e se ne sceglie a colpo d'occhio la copertura migliore, tenendo conto come al solito della possibilità di coprire celle già coperte e condizioni di indifferenza.

Ricordatevi che il numero delle variabili è sempre n meno p , quindi usate questo come esempio, se scrivete un'espressione e vi vengono viene un numero sbagliato di variabili lo capite subito, perché se il numero di variabili è n e l'ordine del rettangolo è p , il numero delle variabili risultante deve essere $N-P$; in questo caso quattro variabili con un ordine due quindi abbiamo due variabili nell'espressione. Questo è il trick che vi deve far capire se sbagliando o meno, è chiaro che due alternative sono veramente alternative se producono espressioni della stessa dimensione, se ci sono due alternative di cui una produce una soluzione con più termini è chiaro che quest'ultima non va bene.

6. Si ripete il passo 5 fino a soddisfare la condizione di copertura. Si scrive infine l'espressione minima $\longrightarrow \underline{acd} + \underline{ac} + (\underline{abd} \text{ oppure } \underline{bcd} \text{ perché equivalenti})$

Perché esistono le forme canoniche di primo e di secondo tipo, con i rispettivi raggruppamenti ?

Questo perché si sceglie se usare un tipo o un altro in base ai valori che prevalgono, se di fatto io noto che ci sono più valori 0 che 1 ovviamente scelgo di raggruppare gli 0, questo perché le espressioni sono equivalenti ma in questa forma sarà minimizzata rispetto all'altra.

In questa mappa ci sono gli 0 sulla prima e sull'ultima colonna, dobbiamo trovare l'implicato primo, quindi dobbiamo ragionare sugli 0, dobbiamo quindi cercare di coprire tutti gli zeri, possiamo quindi creare due raggruppamenti per le due colonne e derivare le formule corrispondenti prendendo ovviamente gli input statici e che quindi non variano, questa volta negati se valgono 1 e diretti se valgono 0 (viene effettuata la somma perché è la seconda forma canonica).

Le condizioni di indifferenza indicano che in le variabili in questa cella, cioè quando i valori delle variabili sono quelli indicati qua, per esempio a pari a zero, b pari a 1, c pari a zero e d pari a 1; vuol dire che non ci interessa quale sia la output, quindi non ci interessa quale sia il circuito che costruiamo per tirare fuori i valori e visto che le condizioni di indifferenza nascono dal fatto che i valori corrispondenti di input non si dovrebbero presentare, per cui non dovendosi presentare l'output non ci interessa che sia corretto, cosa dobbiamo fare quando andiamo a vedere una mappa di karnaugh con le condizioni di indifferenza ?

Li possiamo considerare come uno se stiamo lavorando su una prima forma o come zero se stiamo lavorando sulla seconda forma, quindi li possiamo racchiudere dentro i valori di uno se stiamo lavorando con gli uno, oppure dentro i rettangoli di zero stiamo lavorando lavorando quindi zero, quindi in qualche modo sono come dei jolly, ci sono utili a completare un rettangolo che sia sia di zero che di uno.

Sembrano dei problemi ma in realtà sono delle opportunità.

Allora abbiamo visto quindi con quest'ultimo esempio che le espressioni possono esistere espressioni minime equivalenti.

Dovete distinguere la tecnica dall'applicazione, questa è una tecnica, mentre per l'applicazione sono infiniti gli scenari in cui voi dovete definire un circuito di questo tipo, ci sono altri esempi e questo è l'esempio su cui era andato velocemente prima.

Questo lo avete capito ?

Lo abbiamo visto prima ma lo spiego praticamente, una roba del genere ci dice tutti questi valori sono uno questi sono valori di indifferenza quindi se ragioni in termini di uno come ovvio, non ci sono zeri, questi valori di indifferenza li possono racchiudere e viene il rettangolo più grande possibile di dimensione 16, in cui ovviamente stiamo dicendo che questa funzione darà sempre uno fuori indipendentemente dai valori delle variabili, quindi l'output è identicamente uguale a loro, cioè sostanzialmente è un caso un po speciale.

		cd			
ab		00	01	11	10
	00	0	0	0	1
	01	0	1	-	-
	11	1	1	0	0
	10	1	1	0	0

		cd			
ab		00	01	11	10
	00	0	0	0	1
	01	0	1	-	-
	11	1	1	0	0
	10	1	1	0	0

Altro esempio

	cd			
ab	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	-	1	-	0
10	1	1	-	0

L'espressione minima è:

$$(a + b) \left\{ \begin{array}{l} \bar{b} + d \\ a + d \end{array} \right\} \cdot \left\{ \begin{array}{l} \bar{a} + \bar{c} \\ \bar{c} + d \end{array} \right\}$$

Sintesi minima di un encoder

Si ricorda che nell'encoder, una sola linea di input per volta può assumere il valore 1 e questa determina la codifica binaria in uscita.

L'encoder sostanzialmente cosa fa ?

C'ha tante linee ma solo una linea per volta vale uno mentre le altre valgono zero, la linea che vale uno determina la codifica binaria in uscita; per capirci meglio se in ingresso abbiamo tutte linee che valgono zero in uscita avremmo la rappresentazione binaria del numero 0, se abbiamo in input la prima linea che vale uno e le altre zero è come se stessimo chiedendo al dispositivo combinatorio in questione di generare il numero binario uno, se si attiva la seconda linea in input vogliamo questi generi numero binario due, se si attiva la terza linea il numero binario tre.

Quindi vediamo un pò, 000 in input vuol dire che voglio che in output sia rappresentato il numero binario zero ovvero 00, qui invece mando 001 e attivo la prima linea quindi sto dicendo che voglio modificare il numero 1, quindi in output avrò 01, se qui arriva 010 cioè sto abilitando la seconda linea vuol dire che output si è abilitato il numero 2 quindi 10 e così via.

Gli output tratteggiati in blu nella tabella di verità si ottengono da valori d'ingresso che non dovrebbero proprio verificarsi perché non hanno senso (in questo caso due linee di out o più non possono assumere contemporaneamente il valore 1), per cui vanno considerati condizioni d'indifferenza.

Il numero delle linee di ingresso non è maggiore di quello che serve, però ci sono dei valori che non hanno senso, cioè se io dico che solo una linea di ingresso delle valere uno cosa succede se invece due valori di ingresso valgono 1?

Io mi aspetto che gli input arrivino sempre in un certo formato cioè una sola linea vale 1 e le altre zero e in questo caso mi è chiaro come devo modificare l'output, ma se per qualche motivo mi arrivano gli input due valori uno è chiaro che l'encoder non è in grado di darci una codifica corretta, perché se ci sono due 1 che cosa mando fuori qui?

Non si sa non è previsto, quindi noi dobbiamo comunque creare queste combinazioni perché sono combinazioni che in teoria possono verificarsi, solo per noi andiamo a scrivere qui che questi sono condizioni di indifferenza, cioè noi quale sia l'output in questi casi non ci interessa davvero, perché è responsabilità di chi lo usa mandare degli input che abbiano senso, se poi chi lo usa manda input confusionari siamo liberi in output di mandare quello che ci pare, e queste sono condizioni indifferenza.

È chiaro che queste righe rappresentano valori che non si dovrebbero presentare, ma se si presentassero produrrebbero dei valori di output che a noi non interessano, in altri termini siamo liberi per esempio di dire che quando qui arriva a 011 essendo un valore che non si dovrebbe presentare io tiro poi un output a scelta che non contano.

Dalla tabella di verità dell'esempio, si realizzano due mappe di Karnaugh in quanto sono previsti due output. In seguito, si sceglie la forma canonica con cui lavorare e si formano gli RR. Le espressioni ottenute sono equivalenti e sono:

$$Y_0 = X_2 + X_0$$

$$Y_1 = X_2 + X_1$$

$$Y_0 = X_2 + X_0$$

$$Y_1 = X_2 + X_1$$

Tabella della verità

X_2	X_1	X_0	Y_1	Y_0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	-	-
1	0	1	-	-
1	1	0	-	-
1	1	1	-	-

Data la tabella noi dobbiamo andare a costruire la mappa di Karnaugh, ma in questo caso le mappe sono due, perché come vi ho spiegato noi dobbiamo costruire una mappa per ogni output qui sono due, y con zero ed y con uno, quindi dobbiamo fare la mappa di y con zero e quella di y con uno.

Come si fa la mappa? Essendo tre variabili di input possiamo usare una mappa rettangolare, con le variabili messe una a sinistra e due a destra, andiamo a allocare all'intersezione delle celle i valori, compresi i valori di indifferenza.

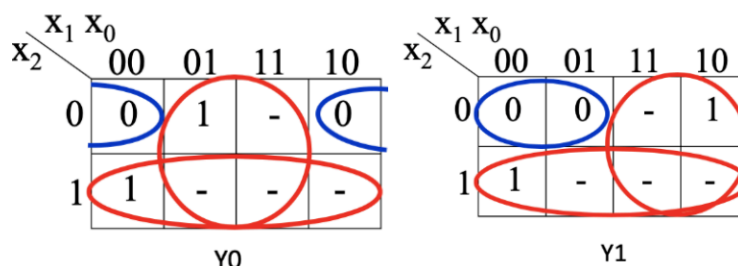
L'ordine con cui mettiamo le variabili di input e quindi formiamo la tabella non ha una regola, l'importante è che siate coerenti.

A questo possiamo andare a identificare su queste due mappe i raggruppamenti o in forma PS o in forma SP, cominciamo con identificare l'espressione minima in somma di prodotti ovvero quella che si basa sugli 1, sono qui identificati dai rossi, è abbastanza semplice perché abbiamo due 1 e tante condizioni di indifferenza; dato quest'1 qual è il rettangolo più grande?

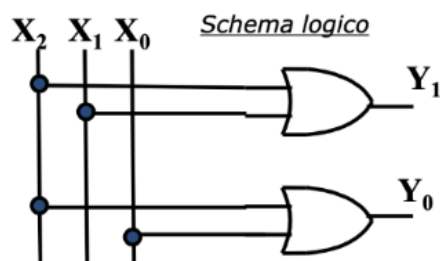
Quello formato da quest'1 e questi tre valori di indifferenza, non possiamo andare oltre, così come questo qua sotto che ci rimane da coprire sarà formato da quest'1 con questi tre adiacenti, quindi andiamo a scrivere quali sono le espressioni minime, ciascuno di questi rettangoli genera una sola variabile visto che stiamo parlando di tre variabili e di rettangolo di dimensione quattro, cioè di ordine due, quindi $3-2 = 1$.

Vediamo adesso cosa succede se lavoriamo con gli 0, si vede già che è lo stesso, ma perché?

Gli 0 sono solamente due e anche in questo caso abbiamo solamente un termine, un termine questa volta chiaramente in somma non prodotto che comprende le due variabili che non cambiano, banalmente è uscita fuori esattamente questa espressione.



Infine, si realizza il circuito che consiste in due porte OR di cui una prende in ingresso X2 e X1 mentre l'altra ha in input X2 e X0.



Una volta che abbiamo ottenuto y con zero ed y con uno dobbiamo scrivere il circuitino equivalente, quindi in entrambi i casi y con uno è la or di due variabili, x due e di x uno, mentre l'altra y con zero prende in or con x due ed x con tre; quindi questo è il circuito che implementa i tre ingressi. Allora l'esempio di un circuito che a questo punto delle vostre conoscenze si deve progettare è il circuito sommatore, è un circuito che prende quattro bit che rappresentano un numero e altri quattro bit che rappresentano un altro numero e produrre la somma di questi due numeri quattro pezzi quindi un circuito ed in output ci porta la somma di questi bit,

lo potremmo trovare in qualsiasi una calcolatrice o in un computer.

Ora se io debbo creare un circuito che somma numeri avente 4/8/16 bit mi devo anzitutto concentrare sul costruire prima il circuito che calcola la somma di due bit tra loro, poi andrò a mettere tutti uno dopo l'altro questi mini circuiti per costruire il sommatore NB.

Full Adder

Il Full Adder è un circuito che prende in input tre variabili (a , b , riporto precedente) e produce in output la somma e il riporto successivo.

C_i	a_i	b_i	C_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Bisogna quindi costruire prima un circuito che in gergo viene chiamato full adder, però ho detto la volta scorsa, un full adder che cos'è?

Un circuito che somma due bit ed il riporto precedente dandoci in output la somma e il riporto successivo, quindi è un circuito che ha tre ingressi che sono i due bit da sommare ed il riporto precedente, in altre parole per sommare due numeri non ci basta sommare i bit ma dobbiamo considerare il riporto precedente, il full adder quindi ha questa tabella di verità che ho disegnato anche la volta scorsa ma velocemente, in cui noi abbiamo tre input tra il riporto precedente e come output produce il riporto successivo e la somma; quindi è chiaro che se sto sommando $0+0$ io riporto precedente era a zero abbiamo che la somma è zero e il riporto pure sarà zero, poi se abbiamo da sommare zero e uno vi riporto precedente la zero avremo la somma sarà uno e non genera riporto e così via, l'unico caso in cui si generano somma uno e riporto uno è quando abbiamo la somma due numeri che valgono uno ed il riporto precedente pari a uno.

Se noi abbiamo questo dispositivo full adder lo possiamo usare per costruire la somma di due generici numeri, immaginate di avere due numeri ciascuno di tre cifre, chiaramente quando andiamo a sommare a zero e b zero il riporto iniziale in input sarà zero e in possiamo dire che è zero, è come se ci fosse un riporto ma è zero, questo genera una somma e anche un riporto successivo, il riporto successivo si usa nella somma delle due cifre successive e così via fino ad arrivare alla somma delle ultime due cifre di ordine $N-1$ cioè in posizione $N-1$ che produce una somma e poi eventualmente produrre un'ulteriore riporto, se ci fossero riportati n numeri da sommare il circuito non sarebbe più di ordine n ma di ordine $n+1$ che potrebbe generare overflow.

Per realizzare un circuito che somma numeri a n bit, però, ho bisogno di un circuito più piccolo che poi sarà messo in serie con altri identici (logica applicata: divide et impera).

Cosa mi serve per costruire un circuito in grado di sommare numeri di dimensione n ?

Mi servono n full adder quindi il primo step è costruire un singolo full adder, allora si costruisce prima il cosiddetto Half adder, un dispositivo combinatorio che somma due bit e produce la somma senza considerare un riporto iniziale, l'half adder funziona se riporto precedente zero e valuta solo i valori di A e B , il valore della somma sarà pari ad uno solo quando uno dei due vale uno, se invece sono entrambi uno o entrambi zero la somma è pari a zero.

Qual è la funzione che vale uno se e solo se è solo uno dei due termini vale uno?

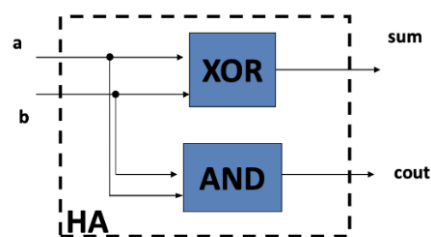
È la xor, noi sappiamo che la xor è un elemento circuitale disponibile o facilmente implementabile, quindi si la somma non è altro che la xor di a e b .

Il Full Adder si ottiene dall'unione di due Half Adder, un circuito che non prevede un riporto precedente in ingresso.

Osservando dalla tavola di verità i casi in cui il C_i è 0, si nota che S_i è pari a 1 solo quando solo uno degli input vale 1, come avviene nella funzione XOR. Invece, il C_o vale 1 solo quando sia a che b valgono 1, come avviene nella funzione AND; si ottiene così un Half Adder.

Invece il riporto successivo vale uno se e solo se entrambi valgono uno, qual è la funzione che restituisce uno se e solo se entrambi gli elementi hanno valore uno?

La and; da qui è molto semplice costruire un full adder, basta utilizzare due half adder, il quale differisce da un half adder perché oltre che ricevere A e B riceve anche un $S(i)$ in che sarebbe di questa ulteriore colonna, quindi questa ulteriore



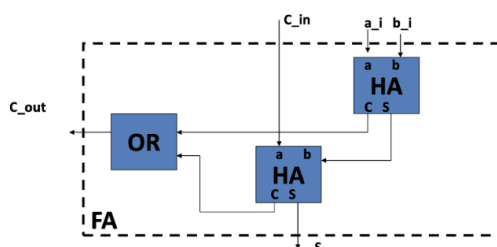
colonna come va a modificare di circuito ?

La prima cosa che facciamo è prendere i due bit e calcoliamo la somma diciamo senza considerare il riporto, poi prendiamo questa somma parziale e lo andiamo a sommare con il riporto, di fatto quello che facciamo è prima sommare le due cifre e quello aggiungiamo un riporto, quindi noi in input successivamente forniamo il riporto precedente calcolata sulle due cifre precedenti, questo determina come risultato la somma e il riporto successivo ma il riporto successivo sarà pari a uno o perché sommando i due numeri si è generato riporto o perché la somma del riporto più la somma genera un rimporto.

La cosa bella della progettazione di calcolatori è che è una progettazione modulare, per cui magari formando vari circuiti semplici componendoli assieme esce qualcosa di estremamente complesso e utile.

Per ottenere il Full Adder, si utilizza un primo HA che riceve in ingresso le variabili a e b e produce in uscita un valore “somma” (S), che fornirà in input al secondo HA, e un valore “riporto” (C), che andrà messo in OR; il secondo HA riceve in ingresso S dal HA precedente e il C_{in} (riporto precedente), che saranno poi sommati, producendo come output la somma finale. Il secondo output di questo HA è il valore “riporto” (C), mandato poi in OR, funzione dalla quale si otterrà il C_{out} (riporto successivo).

Più Full Adder, disposti in maniera modulare, permettono di realizzare n somme. In questo caso il primo C_{in} sarà pari a 0 e ogni “ S ” in output sarà la cifra meno significativa della somma. Inoltre, se operiamo con i numeri naturali ed il $C_{out} = 1$, vuol dire che vi è overflow; invece, se si lavora in complemento a due, si verificherà overflow solo se l’ultimo C_{out} ed il penultimo C_{out} sono discordi (funzione XOR).



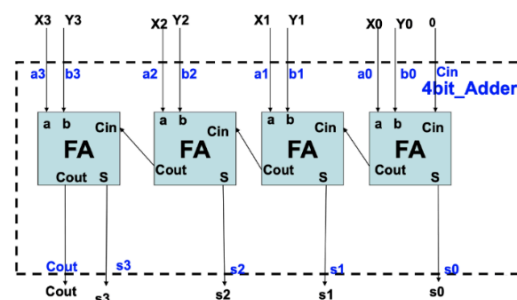
Chiaramente l’esempio che abbiamo fatto è quello di un circuito full adder a 4 bit ma se fosse stato a 16 cosa cambiava ?

Absolutamente niente, basta continuare ad aggiungere blocchetti di full adder a sinistra, ora abbiamo però visto che sommando numeri di dimensione n potrebbe uscire fuori il numero non rappresentabile perché di dimensione $n+1$, se noi usiamo l’adder per sommare due numeri interi positivi a 8 bit usando la notazione

binaria per i numeri interi semplici quella in cui i numeri sono solo positivi, posso rappresentare i numeri da 0 a 255 ora se sommo due numeri che superano il numero massimo ovvero 255 significa che non bastano 8 bit ma ne servono quindi 9, in questo caso il riporto finale pari a uno sarà un campanello di allarme del fatto che io ho superato la soglia massima; tuttavia se sommo due numeri a 8 bit in complemento a due cosa succede ?

Ciò che viene rilevato è il segno dei due numeri, cioè se io sommo due numeri di segno opposto, ovvero un numero con segno iniziale 0 quindi positivo con un numero di segno iniziale 1 quindi di segno negativo non posso andare in overflow, quindi il riporto finale sarà uno comunque ma non avrò overflow; come faccio a capire se la somma dei numeri complemento a due è in overflow ? Abbiamo detto che dobbiamo andare a guardare il riporto qui per ultimo con il l’ultima cifra di output, e abbiamo detto se sono discordi vuol dire che c’è overflow (perché viene applicato lo xor).

Se volessimo modificare il circuito per renderlo in



grado di rilevare l'overflow cosa potremmo fare ?

Dovremo aggiungere una xor che quindi va a vedere se uno dei due vale uno e quindi segnalare l'overflow, quindi diciamo i full adder che lavorano in complemento a due sono una piccola modifica di questo circuito in cui praticamente un'uscita che si ottiene aggiungendo una xor che ci dice se c'è overflow.

APPROFONDIMENTO

ENCODER

Nei circuiti elettronici, gli encoder vengono utilizzati per comprimere più ingressi binari digitali in un numero inferiore di uscite. Anche i convertitori da digitale ad analogico (DAC) e da analogico a digitale (ADC) sono encoder elettronici. Nelle telecomunicazioni, gli encoder vengono utilizzati per convertire i flussi di bit in ingresso in un codice standard per la trasmissione.

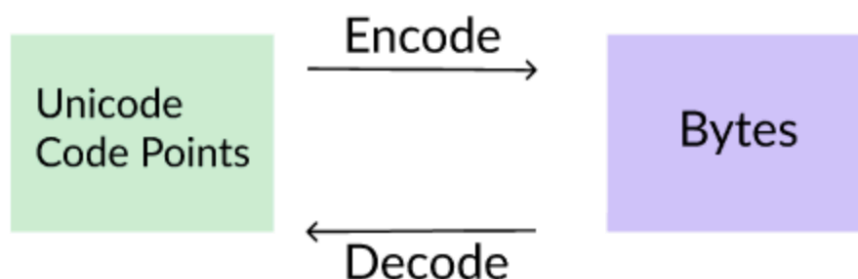
Alcuni trasduttori fungono anche da encoder. Gli encoder rotativi e gli encoder lineari sono esempi di encoder trasduttori. Gli encoder rotativi vengono utilizzati per convertire la posizione angolare di un componente in movimento (ad esempio un albero) e i relativi dettagli in corrispondenti segnali digitali o analogici. Anche i trasduttori lineari svolgono lo stesso tipo di funzione ma su scala lineare. Questi componenti vengono utilizzati in mecatronica e robotica per acquisire informazioni di posizione dei componenti.

Un altro aspetto della codifica è per motivi di sicurezza. Le informazioni, prima della trasmissione o della memorizzazione, potrebbero essere crittografate utilizzando un codificatore, rendendo le informazioni inaccessibili senza un adeguato processo di decodifica; rendendo così sicure le informazioni.

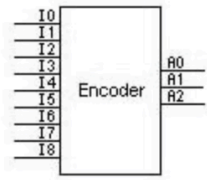
Nella moderna tecnologia multimediale, la codifica viene utilizzata sia nella gestione audio che video. Un codificatore audio può acquisire, comprimere e convertire in un altro formato di dati audio. Un codificatore video può anche eseguire le funzioni di cui sopra per i dati video. Negli ambienti informatici, il software CODEC (Compressor-DECompressor) esegue sia la codifica che la decodifica dei segnali audio – video digitali.

Infine, nelle tecnologie web vengono utilizzati anche encoder per migliorare gli standard di sicurezza. I codificatori di posta elettronica proteggono l'accesso alle email da parte di utenti non autorizzati.

Possiamo dire che, il codificatore (Encoder) converte le informazioni da una forma all'altra (di solito un formato codificato), mentre il decodificatore (Decoder) esegue il processo inverso consentendo il recupero delle informazioni originali.



La funzione logica dell'encoder standard, consiste nel presentare alla sua uscita un determinato codice a seconda dell'ingresso attivato, solo a titolo di esempio, per riportarci al decoder, 2^3 ossia 8 ingressi, avranno 3 uscite, che in codice binario identificano i numeri da 0 a 7. Tra ingresso e uscita non esiste però legame logico come nel decoder perché all'interno dell'encoder esistono delle allocazioni perenni di memoria (memorizzate dal costruttore) tali che il loro numero sia pari alle linee in ingresso (ogni linea attiva individua una locazione di memoria). Se gli ingressi attivati sono più di uno, l'uscita potrebbe assumere una configurazione binaria indesiderata. Per evitare che questo accada, i codificatori in commercio sono "con priorità": se si attiva più di una linea in ingresso, l'uscita assumerà la configurazione associata all'ingresso con priorità maggiore tra quelli attivati. La tabella della verità permette di capire cosa si intende per priorità. La configurazione n°0 presenta l'ingresso I_0 attivato, e i tre bit in uscita codificano la configurazione zero binario. La seconda riga presenta l'ingresso I_1 attivato, e le uscite codificano la configurazione n°1 in binario qualsiasi sia lo stato logico degli ingressi precedenti. L'ingresso I_1 ha quindi maggior priorità rispetto all'ingresso I_0 . Di conseguenza, l'ingresso I_2 ha maggiore priorità di I_1 e I_0 e così via sino all'ultima linea in ingresso.



Conf.	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	A_2	A_1	A_0
	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	X	0	0	1
2	0	0	0	0	0	1	X	X	0	1	0
3	0	0	0	0	1	X	X	X	0	1	1
4	0	0	0	1	X	X	X	X	1	0	0
5	0	0	1	X	X	X	X	X	1	0	1
6	0	1	X	X	X	X	X	X	1	1	0
7	1	X	X	X	X	X	X	X	1	1	1

Esempio di codificatore ad 8 ingressi e 3 uscite con relativa tabella della verità

FULL ADDER

Il full-adder o sommatore completo è un circuito logico caratterizzato da tre ingressi e due uscite. La sua funzionalità è quella di eseguire una somma tra due numeri espressi in formato binario con lunghezza di parola a un bit. È un componente fondamentale dell'elettronica digitale perché, connesso opportunamente con altri full-adder e porte logiche può dare luogo alle unità di elaborazione ALU (Arithmetic Logic Unit) dei processori. I full-adder sono le fondamenta su cui è basata la costruzione di semplici calcolatrici. Il full-adder è costituito dall'insieme di due half-adder e una porta logica OR, opportunamente collegati.

A_n	B_n	C_{n-1}	S_n	C_n
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	0	1
1	1	0	0	1
1	1	1	1	1

Tabella di verità di un full-adder

In logica binaria esegue questa semplice operazione:

$$A + B + C_i = S + C_o$$

dove A e B sono gli operandi, C_i il riporto ($C \rightarrow \text{carry}$) in ingresso della precedente somma e S e C_o sono la somma e il riporto di uscita.

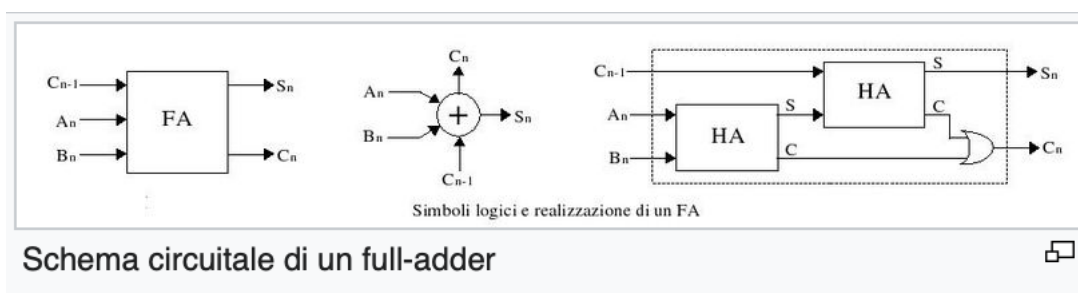
Ogni variabile è un bit (0 oppure 1).

In ingresso sono inseriti i due bit da sommare e l'eventuale bit di riporto; in uscita vengono forniti la somma ed il riporto. Ad esempio, se diamo in ingresso i valori 1 1 0 (1° numero, 2° numero, riporto), il componente restituirà il valore 0 con riporto 1 (corrispondente al valore 10 in base binaria).

$C_i \backslash A_i B_i$					
		00	01	11	10
C_{i-1}	0			1	
	1		1	1	1

$S_i \backslash A_i B_i$					
		00	01	11	10
C_{i-1}	0		1		1
	1	1		1	

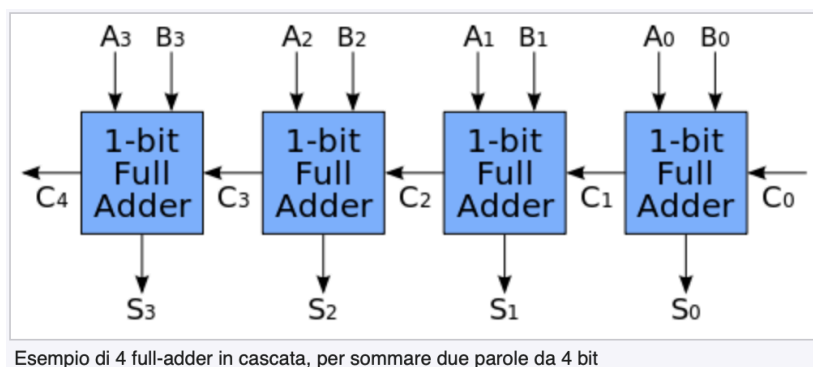
Mappe di Karnaugh del carry out e della somma parziale



FULL ADDER A “N” BIT

La struttura col riporto in ingresso esiste per poter eventualmente collegare un numero "n" full-adder in cascata per poter ottenere Full-adder a "n" bit.

Ottimizzazioni come Kogge-Stone utilizzano strategie di predizione del riporto per ridurre la latenza al costo di occupare maggiore area.



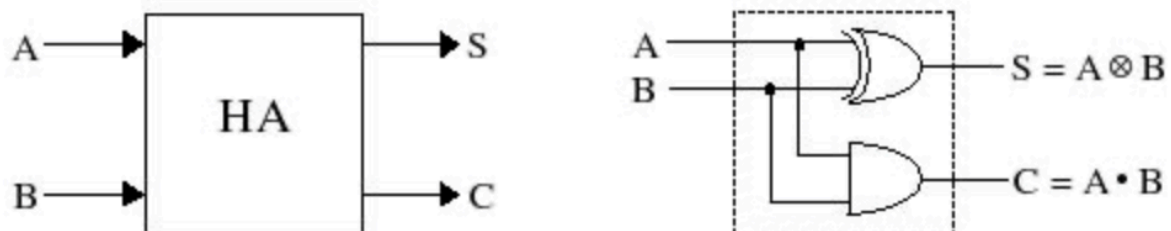
HALF ADDER

L'half adder, detto anche semisommatore, è un componente elettronico digitale che esegue la somma di due bit A e B e la presenta sull'uscita S; inoltre viene calcolato anche il riporto C, senza tener conto però di un riporto precedente. La somma binaria di due bit si calcola allo stesso modo della somma di due numeri decimali, per cui il riporto è generato se e solo se i due addendi sono pari a 1. L'insieme di due half-adder e una porta logica OR, opportunamente collegati, restituisce un full-adder. La seguente tabella della verità mostra la relazione tra gli ingressi e le uscite dell'unità.

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabella della verità del semisommatore

Il circuito è costituito, di conseguenza, da una porta AND e una porta XOR.



Simbolo logico e realizzazione circuitale di un HA